IJE TRANSACTIONS B: Applications Vol. 29, No. 11, (November 2016) 1539-1548

International Journal of Engineering

Journal Homepage: www.ije.ir

Preventing Key Performance Indicators Violations Based on Proactive Runtime Adaptation in Service Oriented Environment

M. Saberi Varzaneh, A. Salajegheh*

Software Engineering and Computer Science of Islamic Azad University, South Tehran Branch, Tehran, Iran

PAPER INFO

Paper history:
Received 12 January 2016
Received in revised form 10 September 2016
Accepted 30 September 2016

Keywords:
Adaptation
Key Performance Indicator
Data Mining
Dependency Analysis
Decision Tree
Service-Oriented Environment
Supply Chain

ABSTRACT

Key Performance Indicator (KPI) is a type of performance measurement that evaluates the success of an organization or a partial activity in which it engages. If during the running process instance the monitoring results show that the KPIs do not reach their target values, then the influential factors should be identified, and the appropriate adaptation strategies should be performed to prevent KPIs violations. In this paper, we propose an integrated monitoring, analysis, prediction and adaptation approach to prevent KPIs violations. We have considered more than one KPI for a specific process and have tried to reach their target values simultaneously by proactive runtime adaptation before the end of the running process. In order to identify the dependency between KPIs and lower-level influential factors, an analysis is done on the data collected from historical process executions. For this purpose, Data Mining techniques have been used. The result is used to predict the KPIs values of the running instance. If KPIs violations are detected, adaptation requirements and adaptation strategies are identified. Since it is possible to define several KPIs for one process, and each has its own importance, so in this paper we tried to satisfy several KPIs target values.

***doi**: 10.5829/idosi.ije.2016.29.11b.07*

1. INTRODUCTION

Key Performance Indicators (KPIs) provide some proper measures to monitor the performance of the system. They check the ability of the system to meet feed quality, quantity and cash flow targets [1]. Since KPIs values are of great importance in a process, reaching their target values is the aim of many systems and organizations. So KPI violation should be predicted and adaptation requirements and adaptation strategies should be identified in order to prevent the violation [2].

In this paper, we have presented a proactive approach that employs runtime adaptation capabilities in order to prevent several KPIs violation and fulfill their target values before the end of the running process. Adaptation is a process of modifying a Service-Based Application (SBA) in order to satisfy new requirements and to fit new situations dictated by the environment [3] and proactive adaptation is to prevent future problems proactively by identifying and handling their sources [3].

In most cases, more than one KPI are defined for a specific process. So, in this paper two KPIs are introduced for our scenario. When more than one KPI is defined, the influence of one KPI should be checked on the others. There may be different states, e.g., the path which one KPI would result in its target value may conflict with other KPIs' paths. In this paper, all the states are considered.

By monitoring the business process while running, and predicting the KPIs values, we have tried to reach their target values. If monitoring shows that KPI targets are not reached, then it is necessary to identify the factors which strongly influence the KPI and cause KPI target violations most often [4]. Sometimes the running process is so complicated that it cannot be easily determined which metrics the KPIs depend on. So, to show the dependencies, machine learning techniques are used. If for a running process instance, KPIs target

*Corresponding Author's Email: *a_salajegheh@azad.ac.ir* (A. Salajegheh)

Please cite this article as: M. Saberi Varzaneh, A. Salajegheh, Preventing Key Performance Indicators Violations Based on Proactive Runtime Adaptation in Service Oriented Environment, International Journal of Engineering (IJE), TRANSACTIONS B: Applications Vol. 29, No. 11, (November 2016) 1539-1548

violations are predicted, adaptation requirements are extracted.

As we have defined more than one KPI for the running process, before extracting adaptation requirements, the KPIs' decision trees should be combined. Then we identify proper adaptation strategies in order to satisfy the adaptation requirements.

The proposed lifecycle contains Modeling, Monitoring, KPIs Dependency Analysis, KPIs Prediction, Instance Trees Combination, Identification of Adaptation Requirements, Selection of Adaptation Strategy and Adaptation Enactment phases. Then by presenting a scenario, producing data and expressing numerous examples, all of the phases have been described.

In the *Modeling* phase, all the information related to lifecycle and scenario has been modeled. In the *Monitoring* phase, all metrics specified in the metric model are monitored. In the *KPIs Dependency Analysis* phase, KPIs models have been constructed based on the data produced and the result is a tree named Dependency Tree. These dependency trees show how the KPIs depend on lower-level process metrics [5]. In the *KPIs Prediction* phase, according to process current conditions and available metrics in the checkpoint, the KPIs predictions are made based on dependency trees, and the result is an instance tree for each KPI. In the *Instance Trees Combination* phase, the rules extracted from instance trees are combined. In the *Identification of Adaptation Requirements* phase, the adaptation requirements are extracted in the form of conjunction of logical predicates from the result of combining the instance trees. In the *Selection of Adaptation Strategy* phase, the property strategies are selected based on adaptation requirements and available adaptation actions in the checkpoint.

2. RELATED WORK

There are several approaches that deal with monitoring and analyzing the business processes in order to prevent KPI violations. They differ mostly in monitoring goals and the analyzing mechanisms. [6] represents an adaptation approach for service based application (SBA) based on a process quality factor analysis. The only KPI in this paper is "*Order Fulfillment Lead Time*". The approach consists of four phases: quality modeling, analysis of influential quality factors, identification and selection of an adaptation strategy and process adaptation. The analysis is based on decision tree algorithms. [7] represents a framework for identifying influential factors of business process performance. The framework uses data mining techniques (J48 and ADTree decision tree algorithms) to construct tree structures for representing dependencies of the KPI, which is defined as "*Order Fulfillment Time*". [4] represents an integrated monitoring, prediction and adaptation approach for preventing KPI violations of business process instances. KPIs are monitored continuously while the process is executed. Additionally, based on KPI measurements of historical process instances, decision tree learning (J48 algorithm) is used to construct classification models which are then used to predict the KPI value. The only KPI of this paper is "*Order Fulfillment Time*". [2] suggested architecture for cross layer adaptation which prevents KPI violation in the service infrastructure (SI) layer which is concerned with a third party. It has concentrated on defining the service level agreement in cross layer adaptation and monitoring the business process. For detecting the KPI violation, decision tree learning is used. The KPI is "*Response Time*". [5] provides a framework for performance monitoring and analysis of WS-BPEL processes, which consolidates process events and Quality of Service measurements. The framework uses machine learning techniques (C4.5 and ADTree techniques) in order to construct tree structures, which represent the dependencies of a KPI on process and QoS metrics.

In our approach, we have proposed a lifecycle which monitors the business process while running and tries to prevent several KPIs violation by runtime adaptation. We have defined two KPIs for a specific process.

By using data mining techniques, specially decision tree algorithms (C5.0 and CRT algorithms), the dependency trees which show how the KPIs depend on lower-level process metrics, are generated.

Since more than one KPI is defined, different states of KPIs safe paths should be checked, and finally a common safe path should be deduced based on KPIs dependency trees in order to reach their target values simultaneously.

Then, we tried to extract adaptation requirements based on the KPIs common safe path, and consequently, proper adaptation strategies are selected in order to satisfy the adaptation requirements.

3. LIFECYCLE

In this section we give an overview of our approach by describing its lifecycle as shown in Figure 1. This lifecycle is created to help business processes to reach their KPIs target values.

The lifecycle consists of the following phases:

- *Modeling*: in this phase (not shown explicitly in Figure 1), metrics model, adaptation actions model, checkpoints model and preferences and constraints model should be created [4].
- *Monitoring*: Information about the running business process and the services it interacts with is collected to monitor KPIs and QoS metrics [7].

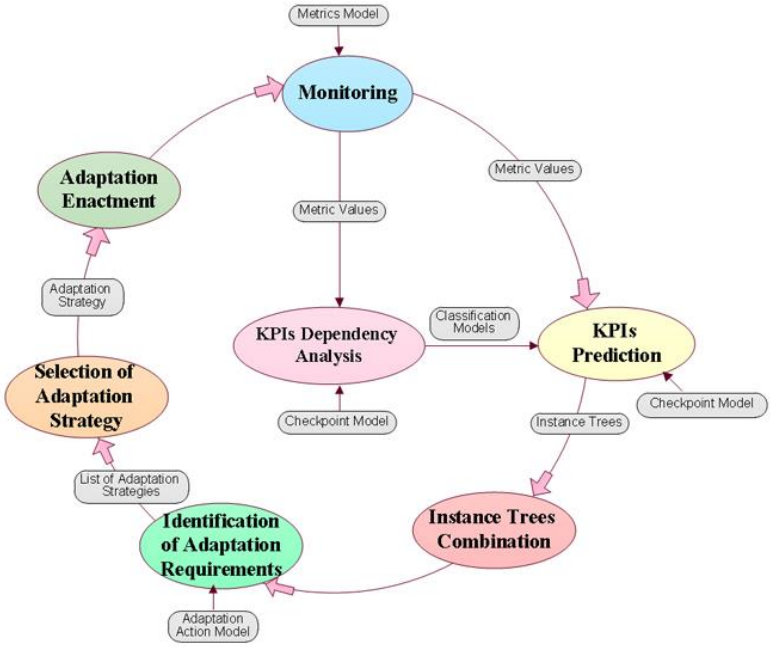

Figure 1. Proposed Lifecycle for Preventing KPIs Violation

- *KPIs Dependency Analysis*: The main idea of dependency analysis is to use historical process instances to determine the most important factors that dictate whether a process instance is going to violate its KPIs or not [5].
- *KPIs Prediction:* when a running process instance reaches a checkpoint, the metric values, which have been measured until the checkpoint for that instance, are gathered and used as the input to the KPIs classification models. The prediction result per KPI is an instance tree (a sub-tree of the original tree), that shows which metrics should be improved in order to reach a specific KPI class [4].
- *Instance Trees Combination*: This paper is focused on several KPIs in a specific process, and the KPIs may have common related metrics, So, before extracting adaptation requirements and strategies, it is necessary to combine the instance trees of KPIs and induce a final rule.
- *Identification of Adaptation Requirements*: from the result of KPIs instance trees combination, it can be identified which metrics should be improved. Therefore, the adaptation requirements can be extracted based on this result.
- *Selection of Adaptation Strategy*: Based on adaptation requirements and available adaptation actions in the checkpoint, the proper strategies can be offered. Each strategy contains a set of adaptation actions which should be enacted to obtain a desirable KPI class.
- *Adaptation Enactment*: the adaptation strategy is enacted by executing the adaptation actions

4. SCENARIO AND KPIs

In this section, we introduce a scenario that is used in the following sections for explaining the approach. The scenario is "*Supply Chain*". Supply chain is a result of linking different operational parts in which suppliers lie at the beginning and customers at the end [8]. Our scenario contains four steps: Indent Reception, Stock Checking, Purchasing and Delivering.

As shown in Figure 2, this process contains Customers, Suppliers and Shippers. At first, the customer sends his request to the office services. The stock keeper checks the items in the stock. If the items exist in the stock, then a shipper is selected and the delivery is done. Otherwise, the purchasing process starts.

♦*Key Performance Indicators*

KPIs are often used in business intelligence (BI) systems to measure the progress of various metrics against business goals.

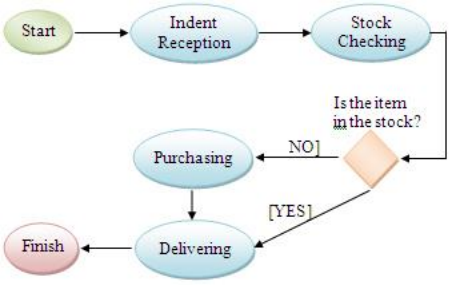

Figure 2. Supply Chain Scenario

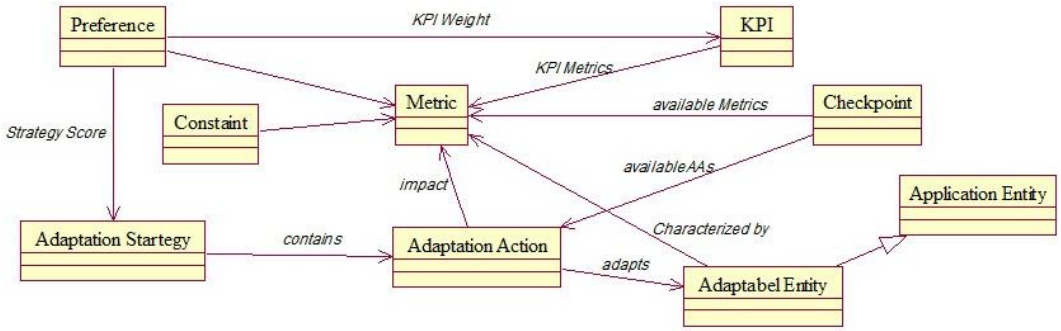

Figure 3. Design-time meta-model

They have become very popular for BI analysis because they provide a quick and visual insight into measurable objectives. KPIs are customizable business metrics that present an organization's status and its trends toward achieving predefined goals in a clear and user friendly format [9].

The KPIs related to the scenario are:

- **KPI1**: The average time between indent reception to the delivery of the ordered items at the customer,
- **KPI2**: The average delivery delay time.

The common metric between these two KPIs is *the shipper*.

5. MODELING

In the Modeling phase, metrics model, adaptation actions model and checkpoints model should be identified, which is done at design time. According to the lifecycle and the scenario, a meta-model (Figure 3) is created that contains all elements such as application entities, metrics, adaptation actions and adaptation strategies. It also incorporates the business logic through relationships and constrains.

This model contains the following classes:

- *Application Entities*: Contains all the entities in the domain; such as the demand item, supplier, shipper and vehicle.
- *Adaptable Entities*: Contains entities which can be adapted and which adaptation actions can be enacted on; such as supplier and shipper.
- *Metrics*: These are lower-level metrics in the domain that are used in the KPIs dependency analysis such as the demand item quantity, availability/lack of items in the stock, supply time, shipping time and vehicle.
- *Key Performance Indicator*: The two KPIs related to the scenario were defined in Section 4.
- *Check Points*: For performing prediction and adaptation, some points in the process should be defined, such as "check in stock".
- *Adaptation Actions*: are a set of actions which can be used for adaptation. Some of these actions are shown in Tables 4 and 5.
- *Adaptation Strategies*: For each adaptation strategy, a set of adaptation actions that can be enacted for this strategy should be defined. One of the domain-independent strategies is "Service Substitution".
- *Preferences and Constraints*: in the Preference Model, metrics, KPIs and strategies can be weighted by assigning a score to each of them. In the Constraint Model, some rules or thresholds can be identified which should never be violated.

6. DEPENDNCY ANALYSIS

The goal of (supply chain) performance management is business process optimization through monitoring and analysis of key performance indicators. By measuring and monitoring metrics against predefined goals, companies can provide added value to large volumes of data generated over time. This type of analysis allows companies to track various metrics at different organization levels and to take timely actions [9].

Sometimes, the running process is so complicated that it cannot be easily determined which metrics the KPIs depend on. So, to show the dependencies, machine learning techniques should be used and also dependency analysis is mapped to a classification problem.

The classification modeling creates a model to map between a set of instances and a set of class labels. It is used to classify new data and is a well-studied technique in data mining and machine learning. There is a range of classification modeling algorithms such as neural network, k-nearest neighbor, support vector machine (SVM), fuzzy rule based classification systems (FRBCSs), decision tree, and Bayesian network [10].

In this paper, we have used data mining techniques and decision tree algorithms.

After a certain number of executed process instances, for each KPI at each checkpoint, a decision tree is trained which helps to understand the

dependencies of that KPI on lower-level metrics, which is called "*Dependency Tree*" [4, 5]. The resulting KPIs dependency trees are used to predict the KPI classes for future process instances.

In order to create the KPI dependency tree, a set of process instances metrics data should have been gathered. For this purpose, according to the scenario, we assumed there are three items I1, I2 and I3 and three suppliers Su1, Su2 and Su3 and three shippers Sh1, Sh2 and Sh3. Then, based on Tables 1, 2 and 3 that show supply time, shipping time and shipper's vehicles, we have produced the *Ordering* and *Delivery* data that are related to the two KPIs.

For each KPI, KPI classes should be specified [4], for example "*Green*", "*Yellow*" and "*Red*" (for classification modeling, continuous values should be converted to discrete values). Then, a *target value function* should be defined to map the KPI values to the classes.

For the Ordering data, according to the expert idea, Order Fulfillment Time should be mapped to three classes:

- Order Fulfillment Time < 14 → "*Green*"
- Order Fulfillment Time ≥ 14 and Order Fulfillment Time ≤ 17 → "*Yellow*"
- Otherwise → "*Red*"

TABLE 1. Supply Time

Supplier	Supply Time		
	Item I1	Item I2	Item I3
Su1	6	8	7
Su2	7	6	8
Su3	6	8	6

TABLE 2. Shipping Time

Shipper	Shipping Time for Item Quantity ≤ 50			Shipping Time for 50 < Item Quantity ≤ 100		
	Item I1	Item I2	Item I3	Item I1	Item I2	Item I3
Sh1	6	5	4	6	8	6
Sh2	4	6	5	7	7	8
Sh3	4	4	7	6	7	7

TABLE 3. Shipper's Vehicles

Shipper	Vehicle
Sh1	Trailer
Sh2	Trailer
	Truck
Sh3	Van
	Truck

For the Delivery data, Delivery Delay Time should be mapped to two classes:

- Delivery Delay Time < 2 → "*Green*"
- Otherwise → "*Red*"

By using SPSS Clementine tool and generating two decision tree algorithms C5.0 and CRT on data, models for two KPIs were created at the "*check in stock*" checkpoint.

Before deploying and utilizing prediction models into production, they must be validated. This is a very important step in the data mining process because we need to know how well models perform against the data [9].

After evaluating the models, model created by C5.0 for KPI1 and that created by CRT for KPI2 were selected (Figures 4 and 5).

As shown in Figure 4, the first breaking parameter for KPI1 is "*order in stock*". For the "*order in stock=true*", it shows that the KPI class is always "*Green*". Also, the KPI2 dependency tree shows that this KPI depends on "*Item Quantity*", "*Ship Time*" and "*Vehicle*" parameters.

The set of metrics which involves in dependency tree are two types: (i) metrics whose values are available at the checkpoint; (ii) metrics whose values cannot be measured until the checkpoint but which are affected by the available adaptation actions of the checkpoint [4]. The first group is used for determining the instance tree, and the latter group is used for extracting adaptation requirements.

For KPI1 dependency tree, "*Order in Stock*" and "*Item Quantity*" are the available metrics, and "*Supply Time*" and "*Ship Time*" are the adaptable metrics.

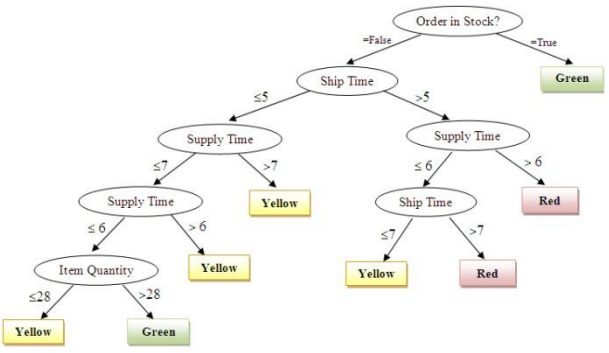

Figure 4. KPI1 Dependency Tree

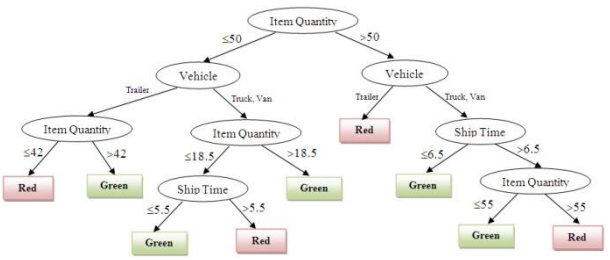

Figure 5. KPI2 Dependency Tree

For the KPI2 dependency tree, the available metric is "*Item Quantity*", and the adaptable metrics are "*Ship Time*" and "*Vehicle*".

7. KPIs PREDICTION and CREATING INSTANCE TREES

Dependency trees can be used for prediction. When the process instance execution reaches a checkpoint, it halts its execution.

The metric values, which have been measured until the checkpoint for that instance (available metrics), are gathered and used as the input to the classification model(s) described in the previous section. This is done at process runtime.

The dependency tree should be traversed breath-first; If the current node corresponds to an available metric, we follow the outgoing branch whose predicate is satisfied by the measured metric value and replace the current node with the target node of that branch; otherwise, if the metric is not available, we leave the node in the tree (and continue with its children until a leaf node is reached). The prediction result per KPI (in general case) is a sub tree of the original tree named "*Instance Tree*" [4].

After obtaining an instance tree for each KPI, we have to merge them and then decide whether adaptation is needed, and if yes, which metrics should be improved and how.

We explain the method by presenting an example. The aim in this step is to predict KPIs classes by using KPI1 and KPI2 dependency trees.

Example (1): The requested item: I1; Item quantity: 65; Order in stock: NO.

For KPI1, since the item does not exist in the stock, the right sub tree of its dependency tree should be deleted. The result is depicted in Figure 6.

Sometimes it is necessary to analyze more. Based on Table 2 (that shows shipping time), the item quantity is one of the influential factors in delivery time. Item quantity in this example is 65, so the shipping time is always more than 5. Therefore, the left sub tree of Figure 6 should be omitted. The final KPI1 instance tree is shown in Figure 7.

For KPI2, item quantity is an available metric and it is 65, so the left sub tree of its dependency tree should be deleted. The result is shown in Figure 8. This figure also shows that when item quantity is 65 and when shipping time is more than 6.5, the final KPI class is "*Red*", so this part should be omitted. The final KPI2 instance tree is shown in Figure 9.

An instance tree shows only those metrics which are yet unknown and also shows how the KPI class of the running instance depends on the metrics affected by available adaptation actions.

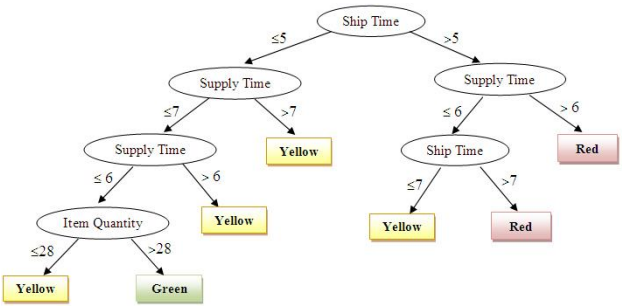

Figure 6. The initial instance tree of KPI1 – example (1)

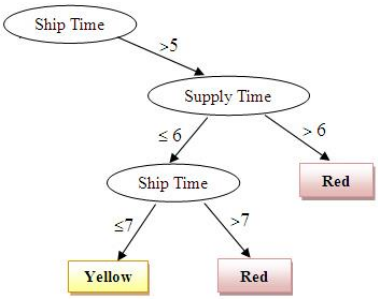

Figure 7. The final instance tree of KPI1 – example (1)

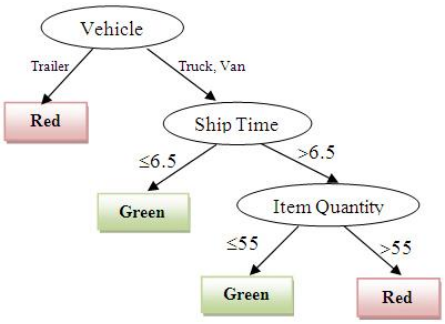

Figure 8. The initial instance tree of KPI2 – example (1)

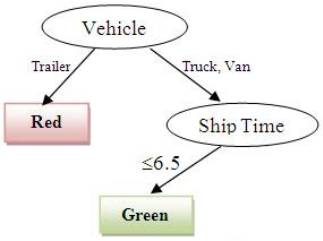

Figure 9. The final instance tree of KPI2 – example (2)

8. INSTANCE TREES COMBINATION

Instance trees are used for extracting adaptation requirements. The novelty of this paper can be considered as defining several KPIs for a specific process and trying to reach their target values simultaneously before the end of the running process.

So, before extracting adaptation requirements, the instance trees should first be combined. The combination of instance trees is done at process runtime.

There are many ways for combining decision trees that are performed on different data but with the same attribute set. For example:

- The learned decision trees are converted to rules and the rules are combined into a single rule [11, 12]
- The learned decision trees are transformed into Fourier spectra and are merged by vector addition in the dual space [13]
- Each tree is represented as a set of decision regions (iso-parallel boxes), then the boxes are intersected efficiently from each of the two trees and finally a tree is induced from the resulting set of boxes [14].

Since we encounter with decision trees having different attribute set, we have used the combinational model to combine them. To combine instance trees, the following steps should be carried out:

- *Step (1)*: Convert instance tree of each KPI into rule sets
- *Step (2)*: Combine rules that lead to desirable classes
- *Step (3)*: Checking the result with the rules that lead to undesirable classes and resolve conflicts if existed.

Step (1): Each path from root to leaf in the tree shows a rule. So by traversing the tree from the root to the leaves, rules can be extracted. These rules are shown as IF_THEN expressions as shown below:

$$\text{If } X_1 \wedge X_2 \wedge \ldots \wedge X_n \text{ Then Class c}$$

X_i is a condition and c is the leaf class.

Step (2): All the rule sets that lead to desirable classes should be checked. For the attributes that have appeared once in the rule sets, the attribute predicate should be copied to the final rule. For the attributes that are repeated for several times, the rules are converted into more sophisticated rules, because more than one KPI target values should be obtained:

- If the attribute test is > then the larger of the two rule values is used (e.g. x>5 and x>8 results in x>8).
- If the attribute test is < then the smaller of the two rule values is used (e.g. x<5 and x<8 results in x<5).
- If the attribute test in one rule is > and in other is < then the rule related to more priority KPI should be assumed.

Step (3): In this step, the resulting rule made from Safe Paths (the paths that lead to desirable classes) should be compared with the rules that lead to undesirable classes. For this purpose, Decision Boundaries method [14] is used. The parameters that are common between safe paths and unsecured paths are shown as decision boundaries. Then, it is checked that whether there are common boundaries between them or not. If yes, the overlapped boundaries are deleted and the range of values is omitted from the final rule.

We explain the method by presenting the examples.

Example (1): The requested item: I1; Item quantity: 65; Order in stock: NO. Desirable classes (*class 1*): {"*Green*" and "*Yellow*"}; Undesirable class (*class 2*): {"*Red*"}.

The influential factors of KPI1 instance tree are "*Ship Time*" (Sh) and "*Supply Time*" (Su). The rule set from KPI1 instance tree are:

- $R_{1\text{-}1}$: Sh > 5 & Su > 6 → *class2*
- $R_{1\text{-}2}$: Sh > 5 & Su ≤ 6 & Sh ≤ 7 → *class1*
- $R_{1\text{-}3}$: Sh > 7 & Su ≤ 6 → *class2*

The influential factors of KPI2 instance tree are "*Vehicle*" (V) and "*Ship Time*" (Sh). Rule set from KPI2 instance tree are:

- $R_{2\text{-}1}$: V = {'Trailer'} → *class2*
- $R_{2\text{-}2}$: V = {'Truck' , 'Van'} & Sh ≤ 6.5 → *class1*

Since "*class1*" refers to desirable class of two KPIs, $R_{1\text{-}2}$ and $R_{2\text{-}2}$ rules should be combined. The result is shown as R(1):

- R(1): V = {'Truck' , 'Van'} & Su ≤ 6 & 5≤ Sh ≤6.5→ *class1*

Now we should check whether R(1) rule overlaps the rules that lead to undesirable class (*class2*) or not. We use the decision regions model. The boundaries of these rules ($R_{1\text{-}1}$, $R_{1\text{-}3}$ and R(1)) are shown in Figures 10 and 11. As shown in these figures, the boundaries do not overlap each other. So R(1) is used as the path that leads to KPI1 and KPI2 desirable classes.

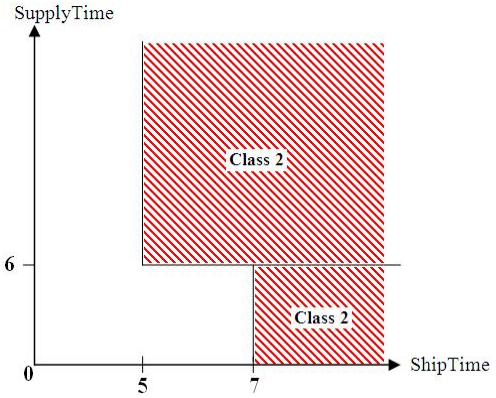

Figure 10. Decision Boundaries of undesirable class of KPI1 – example (1)

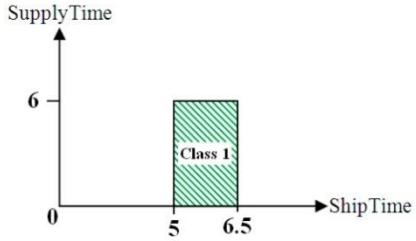

Figure 11. Decision boundaries of final rule R(1)– example(1)

Example (2): In this example, all the paths of instance tree lead to undesirable classes. Consider the desirable class of instance tree shown in Figure 7 is {"*Green*"}. So, the undesirable classes are {"*Yellow*" and "*Red*"}. As shown in this figure, there is no safe path in the tree. So, the combination cannot be done. Section 10 suggests the proper strategies for this case.

Example (3): In this example, the rule sets conflict with each other. Consider the instance trees shown in Figures 7 and 12. The desirable classes are {"*Green*" and "*Yellow*"} and undesirable class is {"*Red*"}. In KPI1 instance tree, to reach the "*Yellow*" class, the shipping time should be more than 5 and less than 7. But KPI2 instance tree shows that if shipping time is greater than 5.5, it leads to "*Red*" class. According to Table 2, all the shipping times are integers and not decimals. So there is no value between 5 and 5.5. Therefore, this case shows a conflict. Section 10 suggests the proper strategies for this case.

9. IDENTIFICATION of ADAPTATION REQUIREMENTS

At the checkpoint, with the KPIs dependency trees, we tried to predict the KPIs classes. With available metrics on the checkpoint, we specified an instance tree for each KPI. By analyzing instance trees, we combined them to induce a final rule that leads to both KPIs desirable classes. Now we identify adaptation requirements which are done at process runtime.

Example (1): the final rule was:

- R(1): V = {'Truck' , 'Van'} & Su ≤ 6 & 5≤ Sh ≤6.5→ *class1*

So, the extracted adaptation requirements are:

- Supply time should be less than or equal to 6
- Shipping time should be greater than or equal to 5 and less than or equal to 6.5
- The shipper should use "*Truck*" or "*Van*" as the vehicle.

Example (2): In this example, no safe path exists, so no requirement is extracted.

Example (3): This example showed the condition that KPI1 and KPI2 instance trees paths conflict with each other. So extracting safe paths and adaptation requirements is not possible.

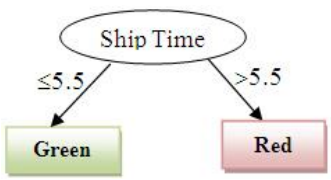

Figure 12. KPI2 instance tree – example (3)

10. ADAPTATION STRATEGY SELECTION

After the requirements have been identified, the next step is to identify adaptation strategies which can be used to satisfy the adaptation requirements. An adaptation strategy consists of a set of adaptation actions which satisfy the metric predicates of an adaptation requirement [4]. Adaptation strategy selection is done at process runtime.

It is necessary to denote the adaptation actions related to the scenario. Based on Tables 1, 2 and 3, we have 27 actions: 9 actions related to supply selection, and 18 actions related to shipper selection. Some of these actions are shown in Tables 4 and 5. Now we recommend strategies for each example.

Example (1): "*Service Substitution*" strategy is used. According to the requirements for this example and Tables 1, 2 and 3, the following adaptation actions are selected:

- Supplier selection: Su1 or Su3
- Shipper selection: Sh3

So we have two strategies:

- Select Su1 as supplier and Sh3 as shipper
- Select Su3 as supplier and Sh3 as shipper.

Example (2): As mentioned, no safe path exists for this example. So, for different processes, the following strategies are recommended:

- If possible, the "Re-execution" strategy [15] is recommended. It means going back in the process to a point defined as safe for redoing the same set of tasks or for performing an alternative path.

"KPI modification" is suggested. When no safe path exists, we can change the range of the KPIs classes' values.

TABLE 4. Supply Service

Adaptation Action	
Type:	Service Substitution
Subject:	Supply Chain: Supplier Service
Service:	Supply I1 by Su1
Impact Model:	Supply Time ≤ 6

TABLE 5. Shipper Service

Adaptation Action	
Type:	Service Substitution
Subject:	Supply Chain: Shipper Service
Service:	Deliver I2 by Sh1 50 < Item Quantity ≤ 100 Vehicle: Trailer
Impact Model:	Ship Time ≤ 8

- For example for KPI1, the range of the "Green" class can be changed to 16:

  Order Fulfillment Time < 16 → "*Green*"

  We should again create the dependency trees. So, it is possible by changing the ranges that some safe paths would be found.
- Business analysts can use the dependency trees to learn about the 'hot spot' of the process, and inform themselves about possible corrective actions if a process underperforms [7]. So, sometimes by "*Modifying Business Process*", we try to reach the KPIs target values in special cases. For example, as KPI1 dependency tree shows, when item exists in the stock, the desirable class is reached ("*Green*"). So, for vital items (items that delaying in supplying them causes serious problems in the process), we must try to always keep them in the stock.

Example (3): This example showed the condition in which KPI1 and KPI2 instance trees paths conflict with each other. In this case, we can use "*Priority Model*". It means assigning a score to each KPI. So, when the rules conflict with each other, we can use the rule of the KPI which has the greater score and has more priority.

In this section, according to the scenario, KPIs dependency trees, KPI instance trees and adaptation requirements, we suggested some strategies, by performing which we try to satisfy KPIs target values.

11. CONCLUSIONS AND FUTURE WORK

Companies often define key performance indicators which are important metrics used to measure the health of the business [9]. Since it is possible to define more than one KPI for each process, we have proposed a novel approach that is used for preventing more than one KPI violations based on runtime adaptation. For describing the phases, we have presented a scenario and two KPIs. By analyzing KPIs data through data mining techniques, their dependency trees that show the relation between lower-level metrics and KPI classes were created. According to the available metrics and KPIs dependency trees, KPIs classes were predicted and instance trees were extracted. By combining the instance trees and merging their rules, a final rule (in general case) that shows the safe path of two KPIs classes was induced. Based on the final rule, the adaptation requirements and the proper strategies were suggested.

For more complex situation, more than one checkpoint might be defined in the process. In this case, the influence of the decision made in one checkpoint should be checked on the other. Or some of the KPIs defined for a process may be more important than the others. In this case, a score number should be assigned to each KPI, and this priority should be assumed in each decision point like instance trees combination phase or selection of adaptation strategies phase. Or, some global constraints may be defined for the process. In this case if some action may violate such a constraint, it should be excluded.

For future work, the following cases are suggested:

- Perform dependency analysis by other algorithms; like Artificial Neural Network (ANN) algorithms and Support Vector Machine.
- Define more than one checkpoint in the process and check the impact of the decision made in one checkpoint on the other.
- Use priority model for KPIs. It means, assigning a score to each KPI, and assume this priority in each decision point.
- Based on this paper's algorithms, suggest a more automated mechanism.

12. REFERENCES

1. Rahmanpour, M. and Osanloo, M., "Resilient decision making in open pit short-term production planning in presence of geologic uncertainty", ***International Journal of Engineering, Transactions A: Basics***, Vol. 29, No. 7, (2016).
2. Parthasarathi, R., Govindasamy, V., Akila, V., Surendar, R., Saranraj, K. and Suresh, S., "Dependency analysis for preventing kpi violation based on decision tree learning", in International Journal of Engineering Research and Technology, ESRSA Publications. Vol. 2, (2013).
3. Zeginis, C. and Plexousakis, D., "Web service adaptation: State of the art and research challenges", ***Self***, Vol. 2, (2010), 5.
4. Wetzstein, B., Zengin, A., Kazhamiakin, R., Marconi, A., Pistore, M., Karastoyanova, D. and Leymann, F., "Preventing kpi violations in business processes based on decision tree learning and proactive runtime adaptation", ***Journal of Systems Integration***, Vol. 3, No. 1, (2012), 3-12.
5. B., W., P., L., F., P., I., B., S., D. and Leymann F., "Monitoring and analysing influential factors of business process performance", ***Institute of Architecture of Application Systems***, (2013).
6. Kazhamiakin, R., Wetzstein, B., Karastoyanova, D., Pistore, M. and Leymann, F., "Adaptation of service-based applications based on process quality factor analysis", in Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, Springer., (2010), 395-404.
7. Wetzstein, B., Leitner, P., Rosenberg, F., Dustdar, S. and Leymann, F., "Identifying influential factors of business process performance using dependency analysis", ***Enterprise Information Systems***, Vol. 5, No. 1, (2011), 79-98.
8. Kamali, H., Sadegheih, A., Vahdat-Zad, M. and Khademi-Zare, H., "Deterministic and metaheuristic solutions for closed-loop supply chains with continuous price decrease", ***International Journal of Engineering-Transactions C: Aspects***, Vol. 27, No. 12, (2014), 1897-1903.
9. Stefanovic, N., "Proactive supply chain performance management with predictive analytics", ***The Scientific World Journal***, Vol. 2014, (2014).
10. Mahdizadeh, M. and Eftekhari, M., "A novel cost sensitive imbalanced classification method based on new hybrid fuzzy cost assigning approaches, fuzzy clustering and evolutionary

algorithms", ***International Journal of Engineering, Transactions B: Applications***, Vol. 28, No. 8, (2015).

11. Hall, L.O., Chawla, N. and Bowyer, K.W., "Decision tree learning on very large data sets", in Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on, IEEE. Vol. 3, (1998), 2579-2584.
12. Hall, L.O., Chawla, N. and Bowyer, K.W., "Combining decision trees learned in parallel", in Working Notes of the KDD-97 Workshop on Distributed Data Mining., (1998), 10-15.
13. Kargupta, H. and Park, B.-H., "A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments", ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 16, No. 2, (2004), 216-229.
14. Andrzejak, A., Langner, F. and Zabala, S., "Interpretable models from distributed data via merging of decision trees", in Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on, IEEE., (2013), 1-9.
15. Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiakin, R., Mazza, V. and Pistore, M., "Design for adaptation of service-based applications: Main issues and requirements", in Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, Springer., (2010), 467-476.

Preventing Key Performance Indicators Violations Based on Proactive Runtime Adaptation in Service Oriented Environment

M. Saberi Varzaneh, A. Salajegheh

Software Engineering and Computer Science of Islamic Azad University, South Tehran Branch, Tehran, Iran

P A P E R I N F O

Paper history:
Received 12 January 2016
Received in revised form 10 September 2016
Accepted 30 September 2016

Keywords:
Adaptation
Key Performance Indicator
Data Mining
Dependency Analysis
Decision Tree
Service-Oriented Environment
Supply Chain

چکیده

شاخص کلیدی عمل‌کرد (KPI) نوعی از اندازه‌گیری عمل‌کرد است که موفقیت یک سازمان یا بخشی از آن را ارزیابی می‌کند. اگر در طول اجرای نمونه فرآیند، نتایج نظارت شده نشان دهند که مقادیر مطلوب KPIها حاصل نمی‌شود، آن گاه بایستی فاکتورهای مؤثر شناسایی و استراتژی‌های مناسب تطبیق به منظور جلوگیری از تخطی KPIها اِعمال گردند. در این مقاله، ما یک روش یک‌پارچه نظارت، تحلیل، پیش‌بینی و تطبیق را به منظور جلوگیری از تخطی چندین KPI پیشنهاد نموده‌ایم. همچنین برای یک فرآیند، بیش از یک KPI در نظر گرفته شده است و سعی کرده‌ایم که قبل از خاتمه فرآیند در حال اجرا و با استفاده از تطبیق پذیری زمان اجرا، به مقادیر مطلوب آنها دست یابیم. به منظور تعیین وابستگی بین KPIها و فاکتورهای مؤثر سطح–پایین، یک تحلیل بر روی داده‌های جمع آوری شده از اجراهای نمونه‌های قبلی فرآیند انجام می‌شود. جهت این کار، از تکنیک‌های داده‌کاوی استفاده شده است. از نتیجه، به منظور پیش‌بینی مقادیر KPIهای نمونه در حال اجرا استفاده می‌گردد. اگر تخطی KPIها تشخیص داده شود، نیازمندی‌های تطبیق استخراج و استراتژی‌های تطبیق انتخاب می گردند. از آنجا که ممکن است برای یک فرآیند، بیش از یک KPI معرفی شود، و هر کدام اهمیت خاص خود را دارند، لذا در این مقاله سعی شده است که به مقادیر بهینه چندین KPI دست پیدا کرد.

***doi**: 10.5829/idosi.ije.2016.29.11b.07*