Improving the Diagnosability of Business Process Management Systems Using Test Points

D. Borrego, Maria Teresa Gómez-López, R.M. Gasca, and R. Ceballos

Dept. de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Sevilla, Spain
{dianabn,maytegomez,gasca,ceball}@us.es

Abstract. The management and automation of business processes have become an essential task within IT organizations, where the diagnosis is a very important issue, since it enables fault isolation in a business process. The diagnosis process uses a set of test points (observations) and a model in order to explain a wrong behavior. In this work, an algorithm to allocate test points is presented, where the key idea is to improve the diagnosability, getting a better computational complexity for isolating faults in the activities of business processes.

Keywords: Process tracing and monitoring, constraint programming, fault diagnosis, fault isolation, test points.

1 Introduction

A business process (BP) is composed of activities which are logically related to achieve a goal. BP management includes concepts, methods and techniques to support the design, administration, configuration, enactment and analysis of BPs [1]. A BP instance is a concrete case in the operational process for a model. If a BP is monitored, some errors can be detected. The diagnosis process detects which tasks are responsible of the incorrect behavior for a business instance.

Fault diagnosis determines why a BP correctly designed does not work as it is expected. Its aim is to identify the reason of an unexpected behavior. The computation is based on observations, which come from the public information existing in the BP, which can be measured by means of test points allocated in certain places of the BP model. In [2] the diagnosis is performed according to the topology of the BP and the relation with the public information monitored.

Test points are control points where it is possible to know data that are available at a moment of the execution. The aim of this work is to improve the diagnosability of a BP by means of the allocation of test points.

The paper is structured as follows: Section 2 raises the allocation of test points in BPs, including three different objectives to achieve. Section 3 shows some experimental results. And finally, conclusions and future work are presented.

2 Allocation of Test Points in a Business Process

The aim of this paper is to apply techniques to allocate test points in BPs.

Definition 1. *Cluster of activities: Being the private information the non-observable information exchanged between the activities, a set of activities T is a Cluster, (i) if it does not exist common private information of any activity of the cluster with any activity outside the cluster, and (ii) if for all $Q \subset T$ then Q is not a cluster of activities.*

Definition 2. *The Diagnosability level is the quotient of the number of faults which can be distinguished each other and the number of all possible faults. Being nAct the number of activities in a BP, the possible faults are initially 2^{nAct} - 1.*

The test points make possible the separation of the activities into different clusters. Also, as it is explained in [4], the computational complexity in the clusters separately is lower than in the whole BP, since the number of possible diagnoses is minor. If all the information within the BP is private, only one fault can be distinguished: the *nAct* activities fail or not. When some test points are allocated and m clusters are obtained, the number of faults that can be distinguished, according to Definition 2, are 2^m - 1.

The proposed algorithm can be configured to achieve three objectives related to diagnosability, presented in next sections. Since the BPs are going to be modelled as CSPs, the transformation into a graph is detailed in the following.

2.1 Improving the Diagnosability by Using Constraint Programming

A CSP consists of $\langle X, D, C \rangle$ where X is a set of n variables $x_1, x_2, ..., x_n$ whose values are taken from finite, discrete domains $D_1, D_2, ..., D_n$ respectively, and C is a set of constraints on their values [3].

A BP can be considered as a directed graph. Its nodes and edges give rise to variables in a CSP (with their domains D):

Variables:
nAct: number of activities
nCon: number of edges
$clusterOfAct_i$: set of *nAct* variables representing the cluster where each activity i is contained ($D : \{0..nAct - 1\}$)
$testPoint_j$: set of *nCon* variables holding the possible new test points in the BP, with possible values *true* (which implies that there must be a test point in a determined connection) and $false$ (the opposite)
nTestPoints: number of allocated test points ($D : \{0..nCon\}$)
nClusters: number of obtained clusters ($D : \{1..nAct\}$)

Each edge gives rise to a constraint within the CSP. As an example, being the connection between two activities A and B the n-th possible test point of the set $testPoint_j$, the constraint added to the CSP would be:

$\texttt{if}(testPoint_n = \texttt{false}) \Rightarrow clusterOfAct_A = clusterOfAct_B$

That constraint means that if the connection between A and B does not count on a test point, they are necessarily in the same cluster. The opposite statement

cannot be asserted, since the existing of the test point cannot imply that A and B are in different clusters since it is possible that they are connected through another path in the graph.

2.2 Objective 1: To Maximize the Number of Clusters Allocating a Fixed Number of Test Points

In order to achieve this objective, the number of test points must be limited to a value t in the CSP. Likewise, the goal is included: taking all the combinations of pairs of values in the set $clusterOfAct_i$, the number of different pairs of values must be maximized, so that the maximum number of activities are placed in different clusters, maximizing the number of clusters obtained.

Being $pairs_{i,j}$ a variable that indicates if each pair of activities i and j are in a different cluster (value 1) or in the same one (value 0).

Constraints: $nTestPoints = t,\ t \in \{1, \ldots, nCon\}$
$\forall i, j \in \{1, \ldots, nAct\}\ (clusterOfAct_i \neq clusterOfAct_j) \leftrightarrow pairs_{i,j} = 1$
Goal: $maximize(\sum_{i=1}^{nAct-1} \sum_{j=i+i}^{nAct} pairs_{i,j})$

The temporal complexity to solve the CSP is exponential for the number of connections. When the BP has a large number of activities, the time needed to solve the CSP makes this solution inappropriate. In order to avoid this problem, the greedy algorithm presented in [4] is used. That algorithm applies the Floyd's algorithm to find the bottlenecks of the BP, which are the most important connections to allocate test points. They are used to select the connections where will be better to allocate test points. Those connections will be the only ones taken into account in the solution of the CSP, improving the computational complexity, although the optimal solution is not guaranteed.

2.3 Objective 2: To Allocate the Minimum Number of Test Points in Order to Obtain a Fixed Number of Clusters

New constraints are added to the initial CSP to establish the number of clusters in a value $numClusters$. The goal is to minimize the number of values equal to $true$ in the set $testPoint_j$. This CSP does not present computational problems.

Constraint: $nClusters = numClusters,\ numClusters \in \{1, \ldots, nAct\}$
Goal: $minimize(nTestPoints)$

2.4 Objective 3: To Minimize the Number of Test Points to Allocate in Order to Obtain Clusters with a Maximum Number of Activities

The initial CSP needs new constraints to limit the number of activities that belong to each cluster to the value $maxNumAct$, and constraints to keep the CSP solver from finding out equivalent solutions (reducing the computational

complexity, since it gets a huge search space reduction). The goal is to minimize the number of test points to allocate, including the following constraints.

The temporal complexity is exponential, so that it is necessary to add some kind of bound to reduce the search space of the variables. In order to get a bound, a new greedy method is used whose solution may not be the optimal solution, but it provides a very useful bound for the number of test points in a linear time that reduces drastically the domain of the variables $clusterOfAct_i$.

Constraints: $clusterOfAct_1 = 0$
$\forall i \in \{0, \ldots, nClusters - 1\}\ occurrences(i, clusterOfAct) \leq maxNumAct$
$\forall i \in \{0, \ldots, nAct - 1\} clusterOfAct_i \leq max(clusterOfAct_j) + 1,\ j \in \{1, \ldots, i - 1\}$
Goal: $minimize(nTestPoints)$

The greedy algorithm is based on the topology of the BPs, using the different control flow patterns existing in the BP model. Since frequently there is a set of branches that form a split and are synchronized by means of a join, it is possible to analyze the processes in a deep way. The splits and joins will enable to divide a BP in different levels. This is, when a single thread of execution splits into branches, and those branches later converge in a join, the activities in those branches are in an inferior level than the activities in the main thread.

Figure 1 shows an example where the BP counts on nine activities. The splits and joins make that the BP has three levels (L1, L2 and L3).

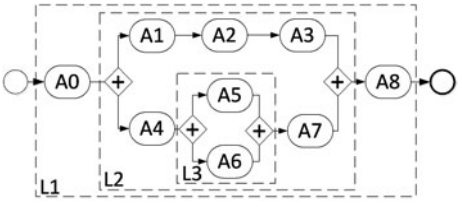

Fig. 1. Business process with three levels

Based on this idea of levels, the greedy algorithm is made up of several steps:

Step 1: Transformation of the BP into a graph and labelling the nodes. The labels of the edges are used to describe if there is a test point in this place. Likewise, each nodes counts on the name of the activity which it is representing, and their labels depend on the levels where they are located: the splits are matched to their corresponding joins, assigning labels from upper to lower levels. The label of the main level (the whole BP) is the string "1". In the rest of levels it is formed by the label of its upper level, concatenating the number of the new level. The numbers in a label indicate the levels where the node is located. At the same time, a tree with the hierarchy of levels is built. Each node of this tree stores the label of a level and the nodes of the graph which are previous and subsequent to that level.

Following with the example in Fig. 1, the different labels assigned to its nodes are shown in Fig. 2(a) and the tree of levels in Fig. 2(b). Level "1" does not have previous and subsequent node, since that level represents the whole BP.

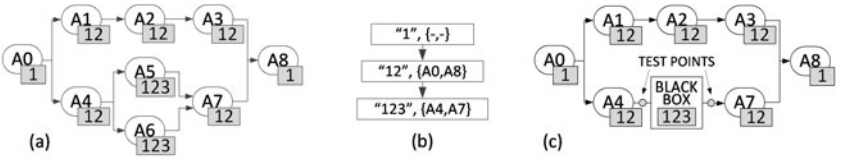

Fig. 2. (a) Graph with labels in the nodes, (b) tree of levels and (c) reduced graph

Step 2: Allocating the test points. Using the tree of levels, this task performs (based on Algorithm 1) a recursive process to allocate the test points. The sentences marked with numbers (1, 2, 3) in the algorithm are detailed in the following.

Algorithm 1. Recursive algorithm to allocate test points

```
if there are more activities in this level than the maximum per cluster then
  if the level is a leaf of the tree of levels then
    (1) allocate test points in the input and outputs of the level in the graph
    (2) allocate test points in the activities of this level in the graph
  else
    for all child c in the tree do
      recursive call: run this algorithm over activities in level c
      if any test point was allocated in level c then
        (3) reduce the graph
      end if
    end for
    (1) allocate test points in the input and outputs of the level
    (2) allocate test points in the activities of this level
  end if
else if this level has exactly the maximum activities per cluster then
  (1) allocate test points in the input and outputs of the level
end if
```

- (1) Allocate test points in the input and outputs of a level: the idea is to isolate the activities of a level from the rest of the activities in the BP. This sentence entails the fact of allocating test points after the previous node and before the subsequent node of the level. For example, if it is necessary to isolate the level "12" of Fig. 2(a), a test point is allocated in the input of the level (output of $A0$) and two in the outputs of the level (outputs of $A3$ and $A7$).
- (2) Allocate test points in a level: either because the level is a leaf or because it has already been isolated, this sentence entails the moment of allocating test points in the activities of a level using the exhaustive CSP explained at the beginning of this subsection.

– (3) Reduction of the graph: once the test points have been allocated in a level, this level must be considered as a *black box* in upper levels. Therefore the graph must be reduced in order to allocate the test points in the whole BP without taking into account the activities of that level.

  In Fig. 2(c) the level "123" has been isolated by means of the test points allocated on it. This level is replaced by a *black box* delimited by test points.

3 Experimental Results

In this section, the temporal complexity of the exhaustive and greedy methods for Objectives 1 and 3 are compared. We present the execution time applied to some BPs with different number of activities (from 5 to 50), which are benchmarks that have been generated to check the Objectives 1 and 3.

Figure 3(a) shows the execution time for the Objective 1. In the chart, the execution time of the exhaustive algorithm and the one that uses the greedy method can be compared. The exhaustive method presents an exponential execution time, whereas the greedy method is polynomial.

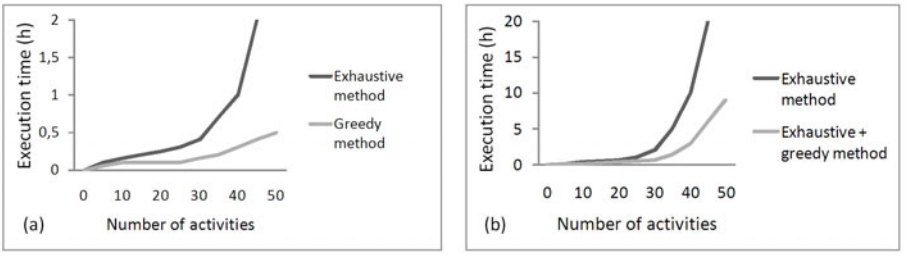

Fig. 3. Execution time for objectives 1 and 3

Likewise, Fig. 3(b) depicts the difference between the execution time spent by the exhaustive and the greedy method for Objective 3. It is possible to see the difference between the exponential execution time for the exhaustive method and the polynomial complexity when the greedy algorithm is used to establish a bound in the number of test points.

4 Conclusions and Future Work

The aim of this work is to improve the diagnosability through applying techniques of allocation of test points, improving the computational complexity of isolating faults in the diagnosis process.

As future work, it is interesting to perform the diagnosis once the test points have been allocated, since they give us additional information that is useful to achieve a more efficient and precise process to find out the minimal diagnosis.

Acknowledgements

This work has been partially funded by Junta de Andalucía by means la Consejería de Innovación, Ciencia y Empresa (P08-TIC-04095) and by the Ministry of Science and Technology of Spain (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

References

1. Weske, M.: Business Process Management. Concepts, Languages, Architectures. Springer, Berlin (2007)
2. Borrego, D., Gasca, R.M., Gómez-López, M.T., Barba, I.: Choreography Analysis for Diagnosing Faulty Activities in Business-to-Business Collaboration. In: 20th International Workshop on Principles of Diagnosis (DX 2009), Stockholm, Sweden (2009)
3. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming. Elsevier, Amsterdam (2006) ISBN 978-0-444-52726-4
4. Ceballos, R., Cejudo, V., Gasca, R.M., Del Valle, C.: A topological-based method for allocating sensors by using CSP techniques. In: Marín, R., Onaindía, E., Bugarín, A., Santos, J. (eds.) CAEPIA 2005. LNCS (LNAI), vol. 4177, pp. 62–68. Springer, Heidelberg (2006)