

## Minimizing test-point allocation to improve diagnosability in business process models



Diana Borrego\*, María Teresa Gómez-López, Rafael M. Gasca

Department of Computer Languages and Systems, University of Seville, Av Reina Mercedes S/N, 41012 Seville, Spain

### ARTICLE INFO

#### Article history:

Received 22 February 2012  
Received in revised form 28 May 2013  
Accepted 28 May 2013  
Available online 12 June 2013

#### Keywords:

Business process improvement  
Diagnosability  
Test points

### ABSTRACT

Diagnosability analysis aims to determine whether observations available during the execution of a system are sufficient to precisely locate the source of a problem. Previous work deals with the diagnosability problem in contexts such as circuits and systems, but no with the adaptation of the diagnosability problem to business processes. In order to improve the diagnosability, a set of test points needs to be allocated. Therefore, the aim of this contribution is to determine a test-point allocation to obtain sufficient observable data in the dataflow to allow the discrimination of faults for a later diagnosis process. The allocation of test points depends on the strategies of the companies, for this reason we defined two possibilities: to improve the diagnosability of a business process for a fixed number of test points and the minimization of the number of test points for a given level of diagnosability. Both strategies have been implemented in the Test-Point Allocator tool in order to facilitate the integration of the test points in the business process model life cycle. Experimental results indicate that diagnosability of business processes can be improved by allocating test points in an acceptable time.

© 2013 Elsevier Inc. All rights reserved.

### 1. Introduction

Nowadays, organizations automate their tasks with business processes (i.e. a set of activities that are performed in coordination in an organizational and technical environment, to jointly realize a business goal (Weske, 2007)) that can be enacted using Business Process Management Systems (BPMS). The fault detection of abnormal behaviours in business processes and the later diagnosis of the responsible for the malfunction are crucial from the strategic point of view of the organizations, since their proper working is an essential requirement. Unexpected faults can provide undesirable halts in the processes, thereby causing cost increase and production decrease. Therefore, to maintain business processes at desirable reliability and production levels, it is necessary to develop automatic techniques to detect and diagnose their faults in order to identify their causes.

Fault diagnosis (hereinafter referred to as diagnosis), is based on observations, which provide information about the behaviour of the process, making possible to discriminate between faults, and hence rendering the business process diagnosable. Without the information obtained from monitoring a process, the faults that occur in runtime cannot be diagnosed since it is not possible to know if the activities composing the business process work correctly.

Therefore, if the information available to perform the diagnosis proceed from few observations, or if observations are not allocated at the most convenient places, it is very difficult to distinguish which parts of the business process are failing. Both the number of observations and the location where they are performed, enable the source of the problem to be precisely located.

Regarding fault handling in business processes, not every kind of fault that can occur is unpredictable. Some faults can be managed at the level of modelling language by catching and handling exceptions, using fault sensors to be fired if a fault occurs during the execution of a monitored activity, thereby detecting the error when it occurs. Nevertheless, according to Han et al. (2009), the existing fault handling mechanism can only detect (identify) the faults which have been pre-defined in standards or by users, but unexpected faults are also the cause of failures in service flows, being necessary an effective diagnosis approach.

The diagnosis process is executed when the actual behaviour of the business process does not correspond to the expected one, being that abnormal behaviour not necessarily perceptible by the use of fault sensors but it may be reported after completion of the execution of the business process (for example, after some customer complaint). In that moment, the diagnosis process is in charge of the isolation of the source of the abnormal behaviour. Therefore, since there is not a single entity that has a global view of information flowing through a business process in runtime, the aim of this approach is to determine the monitoring places in order to guarantee the observability of the data flow at those locations during the execution of a business process instance. This is

\* Corresponding author. Tel.: +34 954 556 234; fax: +34 954 557 139.  
E-mail address: [dianabn@us.es](mailto:dianabn@us.es) (D. Borrego).

performed by the allocation of test points at intermediate places of business processes, and not only at the input and output as it is done by default. A test point can be allocated in the *flow* (sequence flow, conditional flow or default flow according to BPMN 2.0 (OMG, 2011)). This contribution takes into account that not all flows of a business process are available to house a test point, either due to confidentiality, privacy, security, or due to some flows which are not needed to be monitored.

As an example, Fig. 1 shows a business process composed of nine activities. Without any monitoring, the business process presents the minimum diagnosability level, because the lack of information about the data flowing through the process causes that the faulty behaviour of any of the nine activities cannot be discriminated from the rest. That is, given a customer complaint about an abnormal behaviour of the overall process (caused by an unexpected fault), it is not possible to distinguish which activity or activities may be responsible for the problem. Nevertheless, providing the diagnosis system with some information about the inner behaviour of the process by means of the allocation of (at least) one test point in the process in Fig. 1, the diagnosis system will use that information to exonerate some activities from the abnormal behaviour of the process, this way improving the diagnosability level. For instance, the allocation of a test point after the activity  $A_6$  would allow to distinguish whether the abnormal behaviour of the process is caused by some activity in the subset  $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  or in the subset  $\{A_7, A_8, A_9\}$ .

The errors that affect business processes can be derived from different types of faults: (i) business faults, which occur at specific points in a business process because of application issues, for example, because of data content problems. The fault can be the result of a business rule violation, or a constraint violation. For example, invoking a bank service to transfer funds can result in an insufficient-funds fault; (ii) system faults, which occur because of system-related issues, such as the unavailability of a service, or a network failure; and (iii) behavioural faults, which concerns the faults in the model, such as deadlocks and livelocks. The fault diagnosis process is used for the isolation of those activities or sub-processes which are responsible for any incorrect behaviour within the whole process. Since system faults and behavioural faults have already been taken into account in the literature (Varela-Vaca et al. (2011), Baresi et al. (2006), and Eshuis and Kumar (2010); Lin et al. (2002), Van Der Aalst et al. (2011), and Zha et al. (2011), respectively), the present contribution aims to allocate test points for a future diagnosis of business faults. Nevertheless, our proposal can be extended in order to monitor business processes for identifying system faults, in the same way that it is done in the approach by Varela-Vaca and Gasca (2010).

To carry out the idea of isolating business faults, two objectives for the allocation of test points are proposed in this paper: (i) the improvement of the diagnosability level of a business process for a fixed number of test points; (ii) the minimization of the number of test points to allocate for a desired level of diagnosability.

Previous works in the literature deal with the problem of fault detection in business processes (Conforti et al., 2011; Alodib and Bordbar, 2009), and the monitoring of web services or business processes (Yan et al., 2009; Han et al., 2009; Zhang et al., 2009a; Narendra et al., 2008). However, none of these contributions is focused on the analysis of the diagnosability and its improvement. Some other works in the literature address the diagnosability analysis problem (Bocconi et al., 2007; Dressler and Struss, 2003; Travé-Massuyès et al., 2006; Console et al., 2000) for other scenarios.

The paper is organized as follows. Section 2 defines concepts related to diagnosability and introduces the two objectives in greater depth, presenting an example to illustrate the concept of diagnosability in business processes. Section 3 details the

methodology used in the allocation of test points in business processes. Section 4 gives the implementation details and shows experimental results. Section 5 presents an overview of related work found in the literature. And finally, conclusions are drawn and future work is proposed in Section 6.

## 2. Diagnosability of business processes

In order to present the proposal to determine the allocation of test points, it is necessary to introduce the concepts and definitions related to diagnosis and diagnosability. For the sake of clarity, an example is also included.

### 2.1. Main concepts and definitions

The specification of a business process can be viewed from different perspectives (Lanz et al., 2012): (1) the control-flow perspective, which describes the activities of a process as well as their ordering and execution constraints, (2) the data perspective, which connects activities with business and process data, (3) the resource perspective, which provides a link between the process specification and the organizational structure, (4) the operational perspective, which refers to the application services executed in the context of activities, and (5) the temporal perspective, which deals with the temporal properties of the processes.

Although a business process is configured and enacted from a correct model, it may present abnormal behaviour during its execution. This abnormal behaviour is detected because how each activity actually works does not correspond to the expected behaviour, producing wrong data in the data perspective of the process. Those data in the data perspective are dataflow variables, which are read and written by the activities composing the process, and which should be at least partially monitored in order to perform a diagnosis process (cf. Definition 1) to discover the source of the faults.

**Definition 1 (Diagnosis).** A diagnosis of a business process is a particular hypothesis which explains why the current behaviour of the process differs from its expected behaviour.

One of the most used methodology to diagnose classic systems has been model-based diagnosis, which has become the most extensive research area in the diagnosis field. The reasoning is carried out from a model which represents the system to diagnose in an explicit way. A fault exists when the observed behaviour does not correspond with the behaviour expected from the model. This model comes from the knowledge of the system. The component responsible for the fault is identified with a later analysis of the discrepancies.

Model-based diagnosis is based on the comparison between the available observations about the operation of a system (by means of the observation of the dataflow), and the predictions made from the model of the system. The observations indicate how the system is behaving, whereas the model expresses how it should behave during a correct execution.

When a symptom (i.e. a discrepancy between the observed and expected behaviour) is detected, it is deduced that at least one of the components involved in it is not working correctly. The description of the systems, done by the models, uses the relations between inputs and outputs. Most of the approximations for components characterize the diagnosis of a system as a collection of minimal sets of components that fail to explain the observed behaviour (symptoms). That is why it is important to count on a detailed model to determine the diagnosis of a system.

As stated in Bocconi et al. (2007), in order to explain the diagnosability concept, it is necessary to distinguish the notion of fault (i.e. individual state of each activity in a business process), and the

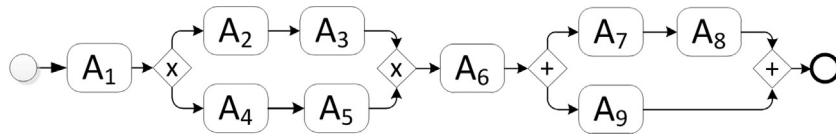


Fig. 1. Example of business process.

notion of fault mode, (i.e. complete state of all the activities in a business process for which observability is done). The individual state of each activity can either be *ok* (working as expected) or *ab* (faulty), such that  $|Act|$  faults can occur in a business process with  $|Act|$  activities. A fault mode is instead a complete specification of *ok/ab* values for all the activities, therefore a system has  $2^{|Act|}$  possible fault modes. Likewise, and following the principle of parsimony (i.e. the simplest of several hypotheses is always the best), the abnormal behaviour of a business process is more likely to be caused by single faults (i.e., adapted to business processes, it implies that only one activity is misbehaving).

The problem is that, counting only on monitoring of the dataflow at the input and output of the processes, it is not possible to discriminate between different fault modes (cf. Definition 2), leading us to a business process where it is not possible to distinguish the activity or activities responsible for the abnormal behaviour. It becomes necessary the monitoring of the dataflow in intermediate flows during the execution of a business process instance. Since the monitoring after each activity is very costly, and sometimes not possible because of privacy and security policies, the monitoring should be performed at strategic places that maximize the diagnostic accuracy, reducing costs.

**Definition 2 (Discriminability).** (Bocconi et al., 2007) Two fault modes are discriminable if their patterns of observable values are disjoint. Then, the discriminability is the capacity to distinguish between fault modes by using the observations available from monitoring.

Therefore, for the model-based diagnosis of business processes to be successful, diagnosability analysis becomes a design-time requirement. The diagnosability level of the business process depends upon the observations of data flowing through the business processes. The observation of data allows a later model-based diagnosis process to determine the source of a business fault which may result in the violation of business policies.

**Definition 3 (Diagnosability level).** Number of fault modes which can be discriminated between them from the observations available from monitoring.

This way, the allocation of test points allows the diagnosability of business processes to be improved, by monitoring the dataflow to locate the source of abnormal behaviour, in accordance with the following definitions.

**Definition 4 (Test point).** A test point is a location within the flow of a business process where the observability of the dataflow is guaranteed.

**Definition 5 (Diagnosability).** (adapted from Console et al., 2000 for business processes) A business process is diagnosable with a given set of test points TP if and only if: (i) for any relevant combination of test-point readings there is only one minimal diagnosis candidate; (ii) every fault of the business process belongs to a candidate diagnosis for certain test-point readings.

## 2.2. Improving the diagnosability level

The diagnosability level of a business process is directly related to the ability to discriminate between faults when they occur. In a

parallel way than in classic diagnosis, several metrics can be used in order to perform a comparison of the diagnosability level of a business process before and after the allocation of test points, such as the longest chain of activities that can be executed in a process instance without any observation of the dataflow between these activities.

However, since the allocation of test points aims to facilitate a later diagnosis process, we opt for the definition of a metric to obtain the number of activities which cannot be discriminated in the worst case, called *#nonIsolatableActivities* (cf. Definition 6). That metric is used as the opposite of the diagnosability level, since the higher the *#nonIsolatableActivities* is, the worse the diagnosability level is, and vice versa.

**Definition 6 (#nonIsolatableActivities).** Metric that represents the size of largest set of activities that cannot be discriminated for any input data combination for a given allocation of test points in a business process.

This idea is illustrated with the example of Fig. 2, where the example of Fig. 1 is monitored by three test points  $tp_1$ ,  $tp_2$  and  $tp_3$ . The example counts on two types of gateways (exclusive and parallel), which influence in the observability of the test points in different ways:

- All parallel branches starting in a parallel gateway are always executed for any process instance reaching the AND split, so that a test point allocated in any of the parallel branches performs its monitoring activity whenever the business process is executed.
- Since only one branch of an exclusive gateway is executed for each process instance, if at least one of the branches does not count on any test point, it is possible that the execution of a process instance flows through one of those branches without test point. This is the worst case regarding diagnosability, and it implies that, existing at least an exclusive branch without test point allocated, a test point in another exclusive branch is not considered to be monitoring the activities outside the exclusive branch where it is allocated when it comes to calculate the diagnosability of the overall business process.

For the example in Fig. 2, Table 1 shows the observations that would be performed by the three allocated test points if some abnormal behaviour is reported. That observations depend on both the activities that participate in the process instance and the faulty activity causing the abnormal behaviour of the overall process, being possible to observe abnormal (*ab*) or correct (*ok*) behaviour of the subset of activities monitored by each test point.

As it is shown in Table 1:

- $tp_2$  and  $tp_3$  always perform monitoring, since both branches of the AND are executed for every possible process instance.

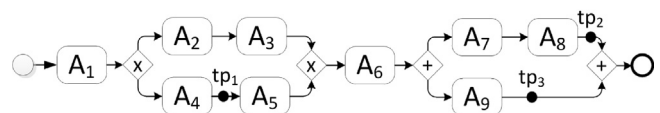


Fig. 2. Example of monitoring of business process.

**Table 1**  
Test point observations.

Activities involved in the process instance	Faulty activity	Observations		
		$tp_1$	$tp_2$	$tp_3$
$\{A_1, A_2, A_3, A_6, A_7, A_8, A_9\}$	$A_1, A_2, A_3$ or $A_6$	–	ab	ab
	$A_7$ or $A_8$	–	ab	ok
	$A_9$	–	ok	ab
$\{A_1, A_4, A_5, A_6, A_7, A_8, A_9\}$	$A_1$ or $A_4$	ab	ab	ab
	$A_5$ or $A_6$	ok	ab	ab
	$A_7$ or $A_8$	ok	ab	ok
	$A_9$	ok	ok	ab

- $tp_1$  only performs monitoring when the process instance includes the branch of the XOR where it is allocated.
- The activities in different branches of the XOR are not taken into account simultaneously, but only if the branch where they are located takes part of the process instance. Then, the number of activities per branch and the test points located within them should be considered when calculating the metric *#nonIsolatableActivities*.

Hence, for the example in Fig. 2, the largest set of activities that cannot be discriminated between them for any input data combination (i.e. *#nonIsolatableActivities*) are  $\{A_1, A_2, A_3, A_6\}$  identified by  $\{tp_2, tp_3\}$  when the process instance is composed of  $\{A_1, A_2, A_3, A_6, A_7, A_8, A_9\}$ . Therefore, *#nonIsolatableActivities* = 4 for this example.

That is, after the allocation of a set of  $n$  test points  $TP = \{tp_1, \dots, tp_n\}$  in a business process, each activity  $A_j$  is monitored by a subset of test points  $TP_k \subseteq TP$ .

In this paper, the flows between activities in the business processes are assumed to be not faulty. Nevertheless, if the flows have to be considered as candidates for the responsibility of any abnormal behaviour, this possibility can easily be modelled by adding a new fictitious component per flow which would be considered during the test-point allocation and later diagnosis process.

### 2.3. Motivating example

This section introduces an example of a business process called *Arrival of a New Employee*, extracted from the Bonita Open Solution documentation (BOS, 2011), shown in Fig. 3. This example is used to illustrate the concept of diagnosability and the allocation of test points in business processes. We use BPMN 2.0 (OMG, 2011) to visualize the model of business processes.

Fig. 3 shows a business process that is executed when a new employee joins the company, and entails the tasks of data notification and of preparation of the new workspace. This process consists of twenty activities (rectangles with rounded corners) and eight gateways or control nodes (diamonds), and a start and end event (circles). A gateway with one incoming edge and multiple outgoing edges is called a split; a gateway with multiple incoming edges and one outgoing edge is called a join.

The activities in the example consume and produce data during the execution of the business process. This flow of data between activities remains unobservable to an external diagnoser until the execution finishes and the end event is reached.

After the execution of an instance of the business process, the observed result may differ from the expected result. This implies that a certain activity or activities within the process present abnormal behaviour. Taking into account that, without the allocation of test points, it is possible to observe data only at the end of the process, and hence the question to answer is: how can the activity or activities which are responsible for the incorrect behaviour of the process be identified? By counting on only the observations at the

end of the process, it is impossible to isolate those activities which cause the fault since they cannot be discriminated from the rest.

For example, if, after the execution of an instance of the process of Fig. 3, the computer that was prepared for the new employee does not include the expected software, was there a mistake in the data notified in the execution of the *Notify hiring management* activity? Or maybe the incorrect behaviour took place at *Requisition software*?

By means of the allocation of test points, we ensure a higher diagnosability of the processes, thereby easing discrimination between activities.

### 2.4. Objectives for the improvement of the diagnosability of a business process

The test points enable the activities of the business process to be discriminated from each other, thereby making it easier to isolate incorrect activities. This is possible since the observations performed by the test points allow us to determine the parts of the business process that are working correctly, and hence only having to perform a later diagnosis process for the remaining parts.

As it has been commented, it is possible to consider that the best solution to the problem is the allocation of test points at every possible location in the business process, thereby isolating each activity from the rest. However, this solution is infeasible due to economic and/or privacy policies:

- The test-point readings take place during the execution of an instance of a business process. In the case of abnormal behaviour of the process, these readings have to be monitored and evaluated by extra human resources, since this IT service is not automated. This issue entails hiring, training and retaining skilled personnel, with the consequent cost increase.
- Likewise, the test-point readings must be handled by a fault-diagnosis system, which is external to the business process. Minimizing the number of transmissions is vital in business processes. Even in the case when the test points are attached to Business Process Management Systems with ample power supply, the reduction in bandwidth consumption may still be a major factor due to the wireless, multi-hop nature of communication and short-range radios.
- Furthermore, not all the possible locations in a business process can include a test point: (i) due to legal regulations (e.g. SOX, HIPAA) and data protection acts, some data flowing through a business process may not be observable due to confidentiality, privacy and security policies; (ii) certain parts of a business process cannot include an observational model which enables the determination of whether the observation performed by a test point at that location differs from the expected observation. Hence, any information collected by the test point is useless from the diagnosis point of view.

Based on these points, the approach presented in this paper applies techniques to allocate test points, considering both the limitation of the number of test points to allocate and the achievement of certain diagnosability level. This brings us to two different objectives for consideration:

- *Objective 1*: If the requirements entail cost limitations, and assuming that the cost of allocating a test point is independent of the location within the business process, this involves a fixed maximum number of test points to allocate. In this case, the objective is to reach the highest possible diagnosability level (minimizing the *#nonIsolatableActivities*) while obeying the economic limitation

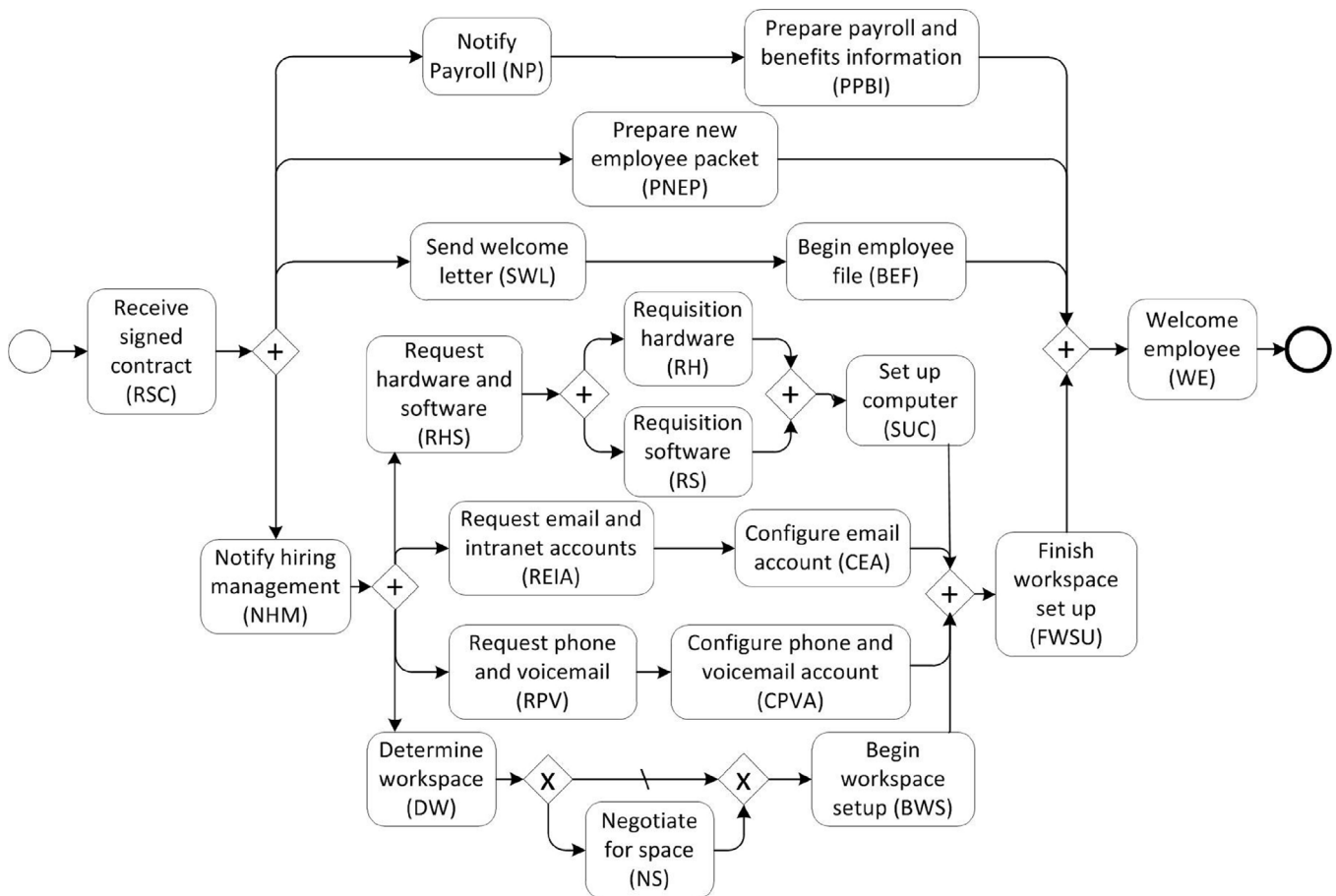


Fig. 3. Running example.

restriction. Therefore, our approach allocates that fixed number of test points and obtains the maximum diagnosability level.

- **Objective 2:** If the requirements entail a certain diagnosability level (for a given  $\#nonIsolatableActivities$ ), the objective becomes the minimization of the number of test points to allocate in order to obtain that level.

### 3. Methodology for objective-driven determination of test-point allocation

A graphical representation of the architecture proposed for the solution of the diagnosability problem is shown in Fig. 4.

This proposal presents a set of steps for the allocation of test points to improve the diagnosability of business processes:

1. Previous to the allocation of test points, business processes have already been modelled using BPMN. The modelling of business processes using this kind of graphical representation entails the automatic generation of .bpmn files which contain the same visual information in XML format. This information is interpreted as input to a Test-Point Allocator to attain an optimal allocation of test points. These test points, together with the initial .bpmn files, provide a monitoring model of the original business processes.
2. This approach starts from these XML files, which contain the information of the overall business process. Each business process in an XML file is denoted by the label `<pools>`, and, within each business process, every activity and gateway has the format:

```
<vertices xmi:type="bpmn:Activity" xmi:id="..."
name="..." outgoingEdges="..." incomingEdges="..."
activityType="..." ... />
```

3. These elements in the XML file can be modelled as a Constraint Satisfaction Problem, with variables and constraints according to the topology. This step is laid out in detail in the following subsection.
4. Depending on which objective is to be achieved, the initial CSP is modified to include some specific constraints and goals. The corresponding algorithm for the allocation of test points is then executed, and the test points are placed.
5. The result is provided as .bpmn files in XML format, and includes new attributes in order to indicate the placement of the test points, thereby attaining monitored business processes by a subsequent diagnosis that would be performed at runtime.

#### 3.1. Solving the diagnosability problem of business processes

As mentioned above, the input to our methodology is a business process model, in BPMN 2.0 format, which has been correctly designed from both control flow and data perspectives.

In order to know the diagnosability level of a business process, by means of the calculation of the  $\#nonIsolatableActivities$  metric, it becomes necessary to count the activities which are monitored by the same subsets of test points allocated in the flows of the process. As a way to perform that counting, we propose to automatically enumerate the activities and control flows (start event, end event, joins and splits) to provide them with a relative ordering w.r.t. the places of the process where some monitoring is performed.

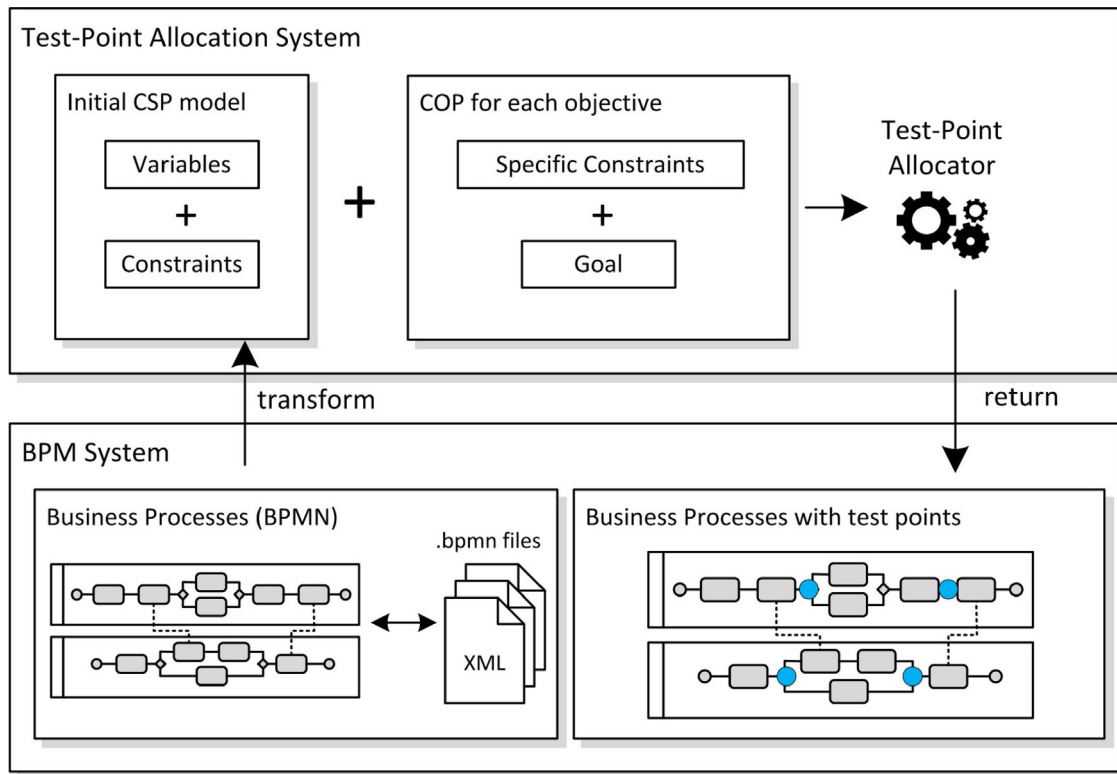


Fig. 4. Graphical representation of the methodology.

As an example, Fig. 5 shows the example in Fig. 1, which does not count on any test point, where the activities have been numbered in accordance with their relative ordering to the output (i.e., in this case, the only place where some monitoring takes place).

The idea of the numbering of the activities and control flows is based on finding the sequence of activities that cannot be isolated because there is not a test point allocated between them. The problem is that, since exclusive and parallel branches can be found, all different possibilities should be analysed. In short, and as shown in Fig. 5, the number of each activity or control flow  $x$  is assigned depending on the elements surrounding  $x$  in the process, as follows:

- Start and end events are always numbered 0.
- The activities in sequential order are enumerated consecutively (e.g. activities  $A_2$  and  $A_3$  in Fig. 5, numbered 2 and 3, respectively).
- The splits are numbered with the same number of their precedent element, since they do not affect to the value of *#nonIsolatableActivities* (e.g. the XOR split in Fig. 5, numbered 1, like  $A_1$ ).
- The first elements in the branches of each XOR are numbered consecutively w.r.t. the XOR split, in order to continue the counting on each branch independently, since only one of the branches would be executed at runtime (e.g.  $A_2$  and  $A_4$ , both numbered 2 because their previous split is numbered 1). This way, each XOR join is numbered with the maximum number between the numbers of the last elements in the branches, such that only the largest set of activities is considered (e.g., the XOR join in Fig. 5 is numbered

3, which in this case is the maximum between the numbers of  $A_3$  and  $A_5$ , both numbered 3).

- As regards the branches of an AND or OR gateway, the first element of one of the branches is numbered consecutively w.r.t. the split, in order to continue the enumeration of previous activities (e.g.,  $A_7$ , numbered 5), and the first elements of the rest branches start a new numbering (i.e. are numbered as 1, like  $A_9$ ). This way, the AND or OR join is numbered as the sum of the numbers of the last elements in each branch, such that the activities in all branches are considered, since all of them would be executed in parallel at runtime and cannot be discriminated if one of them is not working correctly (e.g., the AND join in Fig. 5 is numbered 7, which is the sum of 6 from  $A_8$  and 1 from  $A_9$ ).

Using the numbers assigned in Fig. 5, the metric *#nonIsolatableActivities* can be calculated as the maximum number assigned in the process. In this case, the value of *#nonIsolatableActivities* is 7 indicating that, without the allocation of any test point, there are 7 activities whose abnormal behaviours cannot be discriminated between them.

However, with the allocation of test points in Fig. 2, the numbering of elements changes as shown in Fig. 6, being now *#nonIsolatableActivities* equal to 4.

The main changes in the numbering are: (1) the element after a test point is numbered 1 (e.g.  $A_5$  in Fig. 6) and (2) if exists at least one test point at any branch of an AND or OR, the first

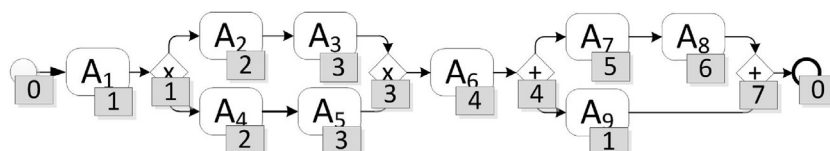


Fig. 5. Example of business process.

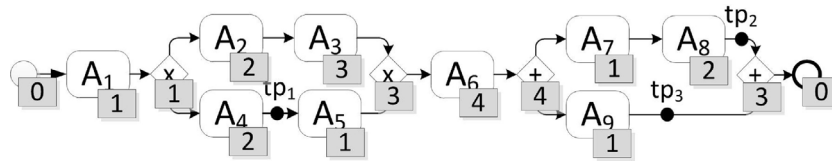


Fig. 6. Example of business process.

element of each branch is numbered 1, since now not all elements in the parallel branches are monitored by the same subset of test points, so that it is not necessary that the first element of one branch continues the previous numbering (e.g.,  $A_7$  is now numbered 1 in Fig. 6). Then, and since the existence of a test point at any parallel branch affects to the numbering of the elements in all branches, the existence of (at least) a test point within an AND or OR is going to be notified by means of labels in the outgoing flows of the corresponding split (as detailed in Section 3.3.2).

Following this idea of the numbering, and in order to obtain the best allocation of the test points, all possible combinations of test-point allocations need to be analysed. For this reason, we could have defined a specific algorithm to optimize the solution. But there is a field in Artificial Intelligence, called Constraint Programming, where this type of problems can be modelled and solved with a set of tools and algorithms to reduce the developing and execution time.

Constraint Programming (Rossi et al., 2006) is based on the resolution of Constraint Satisfaction Problems (CSPs), which seek a consistent assignment of values to variables. A CSP is a triple  $\langle X, D, C \rangle$  where  $X$  is a  $n$ -tuple of variables  $X = \langle x_1, x_2, \dots, x_n \rangle$ ,  $D$  is a corresponding  $n$ -tuple of domains  $D = \langle D_1, D_2, \dots, D_n \rangle$  such that  $x_i \in D_i$ ,  $C$  is a  $t$ -tuple of constraints  $C = \langle C_1, C_2, \dots, C_t \rangle$ . A constraint  $C_j$  is a pair  $\langle R_{S_j}, S_j \rangle$  where  $R_{S_j}$  is a relation on the variables in  $S_j = \text{scope}(C_j)$ . A solution to the CSP  $P$  is an  $n$ -tuple  $A = \langle a_1, a_2, \dots, a_n \rangle$  where  $a_i \in D_i$  and each  $C_j$  is satisfied in that  $R_{S_j}$  holds on the projection of  $A$  onto the scope  $S_j$ .

Our proposal is to transform the business process model and the test-point allocation in a CSP where the best allocation of test points can be obtained in an efficient way.

In order to perform the automatic transformation from BPMN models into CSPs, including the mentioned numbering of elements, we propose an approach based on graph-theory that facilitates the understanding of the problem and the proposed solution, so that each business process model is first translated into a directed graph, called *control flow graph*.

Our proposal is based on the use of a CSP to analyse all the possible allocations of test points. In order to achieve this goal, we use a labelled directed graph to represent the business process, where depending on the location of the test points, the edges that represent the components of the business process will be labelled with numbers related to the distance to a test points. Both the translation into *control flow graph* and the later CSP modelling are detailed in next sections.

### 3.2. Translation of business process models into control flow graphs

As it was aforementioned, the first step in the test-point allocation process is to translate each business process into a *control flow graph*, defined as follows:

**Definition 7 (Control flow graph).** A control flow graph is a tuple  $(N, E)$ , where:

- $N$  is the set of nodes of the graph, composed by the elements of the BPMN model (i.e. activities, XOR splits  $S_x$ , AND and OR splits  $S_{A/O}$ , XOR joins  $J_x$ , AND and OR joins  $J_{A/O}$ , start event and end events);
- $E \subseteq \{(n_i, n_j) \in N \times N : n_i \neq n_j\}$  is a set of directed labelled edges which determine precedence relation between the elements of the process according to the model. The locations of the test points will be represented as true values in the edges.

To simplify the exposition, AND and OR gateways are considered as an only set of splits ( $S_{A/O}$ ) and joins ( $J_{A/O}$ ), since they have the same processing all along the test-points allocation process, since an OR gateway can behave as an AND gateway (all their branches could be executed).

Fig. 7 shows the resulting graph for the running example in Fig. 3.

### 3.3. CSP modelling

Once the *control flow graph* for a business process has been built, an initial CSP is defined from the analysis of the graph in order to collect all necessary information for the CSP solver to allocate the test points.

Likewise, the initial CSP should count on the information to allow the automatic calculation of the *#nonIsolatableActivities* for each attempted test-point allocation. To perform that calculation, the CSP counts on variables and constraints to carry out a numbering of the nodes of the graph. As mentioned, that numbers are intermediate calculations of the metric *#nonIsolatableActivities*, and they depend on the edges where the test points are allocated, on the type of node (activity, split, join, etc.) to number, and on the relative position of the node in the graph w.r.t. the location of the test points which monitor its behaviour.

#### 3.3.1. CSP variables

In order to represent the data managed in the CSP, it is necessary to define the next variables:

- ▷ *nodesNumber*( $n$ ): is a mapping from *node* to *integer* which relates each node of the graph with a number. Depending on the allocation of the test points, the nodes of the graph will be labelled following a set of rules explained in the next subsection.
- ▷ *testPoint*( $e$ ): is a mapping from *edge* to *boolean* which indicates if a test point has been allocated on each edge or not. The possible values for each element are:
  - *testPoint*( $e$ ) = *true* → test point allocated at the edge  $e$ .
  - *testPoint*( $e$ ) = *false* → there is no test point at the edge  $e$ .
- ▷ *label*( $e$ ): is a mapping from *edge* to *boolean* to relate each edge to its label. They are *false* by default, and will be used as a mark to determine the existence of a test point within AND or OR branches.
- ▷ *nTestPoints*: variable holding the number of allocated test points.
- ▷ *forbiddenEdges*: is a set of edges holding the locations that cannot include a test point, so that:
  - $\forall e \in \text{forbiddenEdges}, \text{testPoint}(e) = \text{false}$ .

The roles of these variables are detailed in next subsections by means of indicating their involvement in the constraints of the CSP.

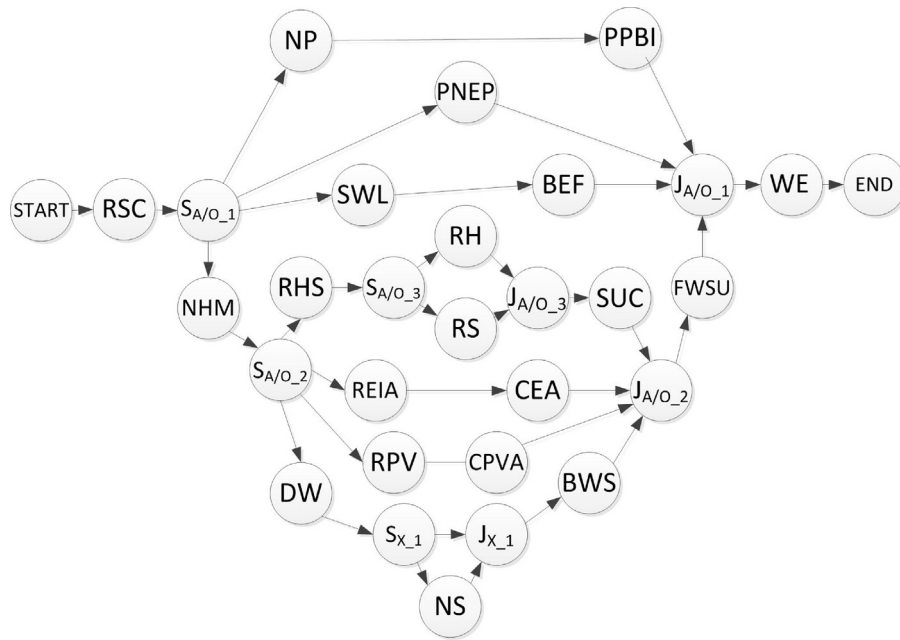


Fig. 7. Control flow graph for the example.

### 3.3.2. Constraints for edge-labelling

In order to model the information regarding the numbering of each node in the graph, indicating the relative order of the nodes, the initial CSP also counts on constraints to perform a runtime labelling of the edges of the graph (variable *label* *i* the CSP), which also depends on the edges where the test points are allocated.

Those labels on the edges are needed to number the nodes within and after the branches of the AND and OR gateways, since they allow the simultaneous execution of parallel branches. Then, it becomes necessary to find out whether there is some test point allocated at any branch to carry out the numbering.

In order to illustrate the idea with an example, Fig. 8(a) shows a part of a business process, composed of four activities  $\{A_a, A_b, A_c, A_d\}$  surrounded by the test points  $tp_x$  and  $tp_y$ . Only counting with those two test points, the process of numbering the nodes in the corresponding graph assigns numbers from 1 to 4 to the activities, as shown in Fig. 8(a), since the abnormal behaviour of the four activities cannot be discriminated between them because all of them are monitored by the same subset of test points ( $tp_y$  in this case). However, if a test point  $tp_z$  is allocated after the activity  $A_c$ , the numbering of the nodes should change as shown in Fig. 8(b). This is due to the allocation of  $tp_z$  changes the discriminability of the activities in the example, being now monitored by different subsets of test points:  $\{A_a, A_c\}$  are monitored by both  $tp_z$  and  $tp_y$ , whereas  $\{A_b, A_d\}$  are monitored only by  $tp_y$ , being now 2 the maximum number of activities not discriminable between them. Therefore, one (ore more) test point allocated at any branch of an AND or OR gateway affects to both the numbering of the nodes in the branches and the numbering of the nodes right after the AND or OR join node.

This way, the label of the edges will be used as a mark to indicate if there is some test point allocated in a branch of an AND or OR gateway or not, and the number assigned to each node in accordance with this information. This is not necessary when it comes to number the nodes in a XOR gateway, since it only allows the execution of one branch at a time, so that a test point allocated in one branch only affects to the numbering of the nodes in that branch.

Thus, a boolean label is assigned to each edge  $e$  (referred to as  $label(e)$ ) during the allocation process (as the CSP solver goes along), considering certain rules.

For each node  $n$  in  $N$ , and being  $E' \subseteq E$  the set of outgoing edges of  $n$ :

- If  $n$  is not an AND or OR split, all the edges in  $E'$  are labelled as *false*,

$$n \notin S_{A/O} \rightarrow \forall e \in E' : label(e) = false$$

- If  $n$  is an AND or OR split, considering  $N' \subseteq N$  the set of nodes in the parallel branches beginning in  $n$ , and being  $e'$  one of the edges in  $E'$ :

- the remaining edges in  $\{E' - e'\}$  are labelled as *true*;
- the label for  $e'$  will be *true* iff there is at least one test point allocated in one of the flows which compose the parallel branches of the AND/OR beginning in  $n$  (so that at least one pair of different nodes in  $N'$  can be discriminated between them in accordance with the monitoring of the allocated test points), and *false* otherwise.

The edge  $e'$  can be any of the outgoing edges of  $n$ , since  $E'$  is a set, and its label will affect the numbering of every node within the AND or OR branches and the nodes after the join, not depending on the edge where the label is assigned.

In a formal way, being  $N' \subseteq N$  the set of nodes in the parallel branches beginning in  $n$ , and considering that for each pair of nodes  $n_i, n_j \in N, n_i \neq n_j$  the function  $disc(n_i, n_j)$  is true iff  $n_i$  and  $n_j$  are discriminable,

$$n \in S_{A/O} | e' \in E' \rightarrow ((\forall e \in \{E' - e'\} : label(e) = true) \wedge (label(e') = true \leftrightarrow \exists n_i, n_j \in N' | n_i \neq n_j : disc(n_i, n_j) = true))$$

For the example of control flow graph in Fig. 7, once the test points are allocated, all the edges in the graph will be labelled as *false* by default, but the outgoing edges of each AND or OR split, which are labelled in accordance with the previous expression. For this example, the edges with different treatment are:

- Edge  $(S_{A/O_1}, NP)$  for the split named  $S_{A/O_1}$  in Fig. 7. This edge will be labelled as *true* iff the test-point allocation enables the discrimination of the node RSC (previous node to  $S_{A/O_1}$ ) from at



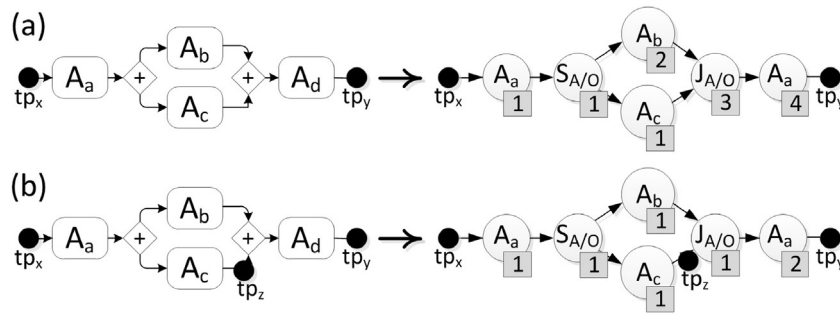


Fig. 8. Partial example of a monitored business process.

least one of the nodes within the AND/OR parallel branches: NP, PPBI, PNEP, SWL, BEF, NHM,  $S_{A/O.2}$ , RHS,  $S_{A/O.3}$ , RH, RS,  $J_{A/O.3}$ , SUC,  $J_{A/O.2}$ , REIA, CEA, DW, RPV, CPVA,  $S_{X.1}$ , NS,  $J_{X.1}$ , BWS, FWSU.

That is,  $label((S_{A/O.1}, NP)) = true$  iff there is a test point at least at one of these edges: (RSC,  $S_{A/O.1}$ ), (NP, PPBI), (PPBI,  $J_{A/O.1}$ ), (PNEP,  $J_{A/O.1}$ ), (SWL, BEF), (BEF,  $J_{A/O.1}$ ), (NHM,  $S_{A/O.2}$ ), (RHS,  $S_{A/O.3}$ ), (RH,  $J_{A/O.3}$ ), (RS,  $J_{A/O.3}$ ), (SUC,  $J_{A/O.2}$ ), (REIA, CEA), (CEA,  $J_{A/O.2}$ ), (RPV, CPVA), (CPVA,  $J_{A/O.2}$ ), (DW,  $S_{X.1}$ ), (BWS,  $J_{A/O.2}$ ) or (FWSU,  $J_{A/O.1}$ ).

- Edge ( $S_{A/O.2}$ , RHS) for the split named  $S_{A/O.2}$  in Fig. 7. This edge will be labelled as *true* iff the test-point allocation enables the discrimination of the node NHM (previous node to  $S_{A/O.2}$ ) from at least one of the nodes within the AND/OR parallel branches: RHS,  $S_{A/O.3}$ , RH, RS,  $J_{A/O.3}$ , SUC, REIA, CEA, DW, RPV, CPVA,  $S_{X.1}$ , NS,  $J_{X.1}$ , BWS.

That is,  $label((S_{A/O.2}, RHS)) = true$  iff there is a test point at (at least) one of these edges: (NHM,  $S_{A/O.2}$ ), (RHS,  $S_{A/O.3}$ ), (RH,  $J_{A/O.3}$ ), (RS,  $J_{A/O.3}$ ), (SUC,  $J_{A/O.2}$ ), (REIA, CEA), (CEA,  $J_{A/O.2}$ ), (RPV, CPVA), (CPVA,  $J_{A/O.2}$ ), (DW,  $S_{X.1}$ ) or (BWS,  $J_{A/O.2}$ ).

- Edge ( $S_{A/O.3}$ , RH) for the split named as  $S_{A/O.3}$  in Fig. 7. This edge will be labelled as *true* iff the test-point allocation enables the discrimination of the node RHS (previous node to  $S_{A/O.3}$ ) from at least one of the nodes within the AND/OR parallel branches: RH, RS.

That is,  $label((S_{A/O.3}, RH)) = true$  iff there is a test point at (at least) one of these edges: (RHS,  $S_{A/O.3}$ ), (RH,  $J_{A/O.3}$ ) or (RS,  $J_{A/O.3}$ ).

### 3.3.3. Constraints for node numbering

As it was aforementioned, the goal of the initial CSP is to model the relation between the test-point allocation, the *#nonIsolatableActivities* of the overall business process, and its topology. These constraints aims to perform a numbering of the nodes in the graph (variable *nodesNumber* of the CSP). The idea of this numbering of the nodes is based on a combination among the distance of a node from a test point and the topology of the business process, so that the *#nonIsolatableActivities* can be calculated from this modelling.

The key idea for the modelled CSP is to establish values to the activities, using the mapping *nodesNumber*, to determine a relative ordering with respect to each allocated test point. Those values are used to calculate the maximum number of activities whose abnormal behaviours cannot be discriminated between them, i.e. the *#nonIsolatableActivities* of the business process.

The constraints to establish the numbering of each node is different depending on the type of nodes that are surrounding it, and on whether a test point is allocated in the flow previous to the node. This way, if a node  $n_i$  is preceded by a test point  $tp_j$ , its value is equal to 1 due to  $n_i$  is just following  $tp_j$ , so that it becomes necessary to initialize the numbering in order to start a new relative ordering w.r.t. the following test point. In other case, the value of  $n_i$  would be calculated according to the values of the nodes preceding  $n_i$ , as it is detailed in the following and shown in Table 2.

- **Case 1:** The value of the start and end nodes (START and END) is always equal to 0.
- **Case 2:** Two nodes  $n_x$ ,  $n_y$ , where  $n_y$  is an activity, connected through an edge  $e_j$ .
  - If the test point  $tp_j$  is allocated in  $e_j$  (i.e.  $testPoint(e_j) = true$ ), the value of  $n_y$  is equal to 1, since  $n_y$  is the first node after  $tp_j$ .
  - If the label of  $e_j$  is true (i.e.  $label(e_j) = true$ ), the value of  $n_y$  is equal to 1, since  $n_y$  is located in a branch of an AND or OR that does not need any special consideration.
  - Otherwise, the value of  $n_y$  is calculated as  $n_y = n_x + 1$ , since  $n_y$  and  $n_x$  are hence monitored by the same test points, and they cannot be discriminated between them.
- **Case 3:** Two nodes  $n_x$ ,  $s$ , where  $s$  is a split, connected through an edge  $e_j$ .
  - If the test point  $tp_j$  is allocated in  $e_j$  (i.e.  $testPoint(e_j) = true$ ),  $s$  is numbered 0, since  $s$  is the first node after  $tp_j$ , but it is not taken into account for the calculation of *#nonIsolatableActivities* since it is not an activity.
  - If  $tp_j$  is not allocated, the value of  $s$  is calculated as  $s = n_x$ , since  $s$  and  $n_x$  are observed by the same set of test points.
- **Case 4:**  $J_{a/o}$  is an AND or OR join, with nodes  $n_y$ ,  $n_w$ ,  $n_z$ , ... as sources.
  - Being *not\_monitored\_nodes* the nodes in  $n_y$ ,  $n_w$ ,  $n_z$ , ... that do not count on a test point allocated at their outgoing edges  $e_j$ ,  $e_k$ ,  $e_l$ , ..., the value of the join  $J_{a/o}$  is calculated as  $J_{a/o} = \sum not\_monitored\_nodes$ . This way, the node  $J_{a/o}$  collects the quantity of nodes that have not been monitored by any test point yet.
- **Case 5:**  $J_x$  is a XOR join, with nodes  $n_y$ ,  $n_w$ ,  $n_z$ , ... as sources.
  - Being *not\_monitored\_nodes* the nodes in  $n_y$ ,  $n_w$ ,  $n_z$ , ... that do not count on a test point allocated at their outgoing edges  $e_j$ ,  $e_k$ ,  $e_l$ , ..., the value of the join  $J_x$  is calculated as  $J_x = \max(not\_monitored\_nodes)$ . This way, the join  $J_x$  collects the quantity of nodes that have not been monitored by any test point yet in the worst case.




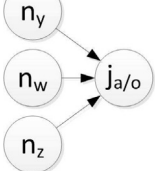
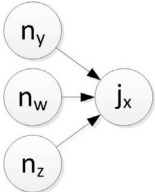
As an example, Table 3 presents some constraints for some different types of nodes in the control flow graph in Fig. 7.

### 3.3.4. Calculation of #nonIsolatableActivities

As it was aforementioned, the aim of the nodes numbering is to calculate the *#nonIsolatableActivities* of the overall business process. Those numbers are assigned to the nodes in order to count how many activities are monitored by the same sets of test points, that is, how many nodes cannot discriminated between them in accordance with the allocated test points. This way, the value of the *#nonIsolatableActivities* of the process can be calculated as the maximum number between the nodes numbering, since that value is the maximum number of activities monitored by the same set of test points.

$$\#nonIsolatableActivities = \max(nodesNumber(x)), \quad x \in N$$

**Table 2**  
Constraints for each type of node.

Case	Control flow	Constraints to model the node numbering
1		$nodesNumber(START) = 0$ $nodesNumber(END) = 0$
2		$(testPoint(e_j)    label(e_j)) \rightarrow nodesNumber(n_y) = 1$ $\neg(testPoint(e_j)    label(e_j)) \rightarrow nodesNumber(n_y) = nodesNumber(n_x) + 1$
3		$(testPoint(e_j)    label(e_j)) \rightarrow nodesNumber(s) = 0$ $\neg(testPoint(e_j)    label(e_j)) \rightarrow nodesNumber(s) = nodesNumber(n_x)$
4		$nodesNumber(j_{a/o}) = 0$ $(testPoint(e_j) = false) \rightarrow nodesNumber(j_{a/o}) = nodesNumber(j_{a/o}) + nodesNumber(n_y)$ $(testPoint(e_k) = false) \rightarrow nodesNumber(j_{a/o}) = nodesNumber(j_{a/o}) + nodesNumber(n_w)$ $(testPoint(e_l) = false) \rightarrow nodesNumber(j_{a/o}) = nodesNumber(j_{a/o}) + nodesNumber(n_z)$
5		$nodesNumber(j_x) = 0$ $testPoint(e_j) = false \rightarrow nodesNumber(j_x) = nodesNumber(n_y)$ $testPoint(e_k) = false \rightarrow j_x = \max(nodesNumber(j_x), nodesNumber(n_w))$ $testPoint(e_l) = false \rightarrow j_x = \max(nodesNumber(j_x), nodesNumber(n_z))$

In order to illustrate the aim of the node numbering, Fig. 9 shows the numbers which result to the application of the constraints in Table 3 without the allocation of any test point ( $nTestPoints = 0$ ). It is possible to notice that the maximum number is 20 in node WE (i.e.  $\#nonIsolatableActivities = 20$ ), which corresponds to the number of activities in the original business process in Fig. 3 since, without the allocation of any test point, none activity is discriminable from the rest.

This initial CSP is the base for the two objectives to reach, and its constraints are common to the two objectives achieved by the solution proposed in this paper. However, the initial CSP needs to be enriched with specific constraints and an objective function depending on the objective to achieve. Hence, a Constraint Optimization Problem (COP) is attained, which can be defined as a regular CSP whose goal is to find the optimal solution in accordance with the determined objective function. The two objectives

and their corresponding configurations are given in detail in the following subsections.

#### 3.4. Objective 1: maximization of the diagnosability level with the allocation of a fixed number of test points

Beginning with the initial CSP, it is necessary to add a new constraint as well as the goal to be achieved for this specific objective. This new information is also generated and included in the initial CSP in an automatic way.

More precisely, the new information to be added is:

- The number of test points must be limited to a value  $t$  indicated by the user or the problem specification. That is, the number of true values in the mapping  $testPoint$  has to be equal to the number of test points to be allocated, hold in the variable  $nTestPoints$ , which remains constant for this objective. If this value  $t$  is greater than or equal to the maximum number of existing flows for the allocation of test points, it is not necessary to execute any algorithm, since a test point will be allocated at each possible location of the business process.
- The goal is included in the CSP: the objective function becomes the minimization of the  $\#nonIsolatableActivities$  of the business process, thereby obtaining the maximum diagnosability level.

**Table 3**  
Some constraints for the example in Fig. 3.

Node START	$nodesNumber(START) = 0$
Node RSC	$nodesNumber(RSC) = 1$
Node $S_{A/O.1}$	$(testPoint((RSC, S_{A/O.1}))    label((RSC, S_{A/O.1}))) \rightarrow nodesNumber(S_{A/O.1}) = 0$ $\neg(testPoint((RSC, S_{A/O.1}))    label((RSC, S_{A/O.1}))) \rightarrow nodesNumber(S_{A/O.1}) = nodesNumber(RSC)$
Node NP	$label((S_{A/O.1}, NP)) = (testPoint((RSC, S_{A/O.1}))    testPoint((NP, PPBI))    testPoint((PPBI, J_{A/O.1}))    testPoint((PNEP, J_{A/O.1}))    testPoint((SWL, BEF))    testPoint((BEF, J_{A/O.1}))    testPoint((NHM, S_{A/O.2}))    testPoint((RHS, S_{A/O.3}))    testPoint((RH, J_{A/O.3}))    testPoint((RS, J_{A/O.3}))    testPoint((SUC, J_{A/O.2}))    testPoint((REIA, CEA))    testPoint((CEA, J_{A/O.2}))    testPoint((RPV, CPVA))    testPoint((CPVA, J_{A/O.2}))    testPoint((DW, S_{X.1}))    testPoint((BWS, J_{A/O.2}))    testPoint((FWSU, J_{A/O.1})))$ $(testPoint((S_{A/O.1}, NP))    label((S_{A/O.1}, NP))) \rightarrow nodesNumber(NP) = 1$ $\neg(testPoint((S_{A/O.1}, NP))    label((S_{A/O.1}, NP))) \rightarrow nodesNumber(NP) = nodesNumber(S_{A/O.1}) + 1$

#### Constraint

$nTestPoints = t$

#### Goal

minimize( $\#nonIsolatableActivities$ )

As an example, for the control flow graph in Fig. 7, and setting the number of test points to allocate to  $nTestPoints = 3$ , our approach allocates those test points as it is shown in Fig. 10. All the activities in a business process are monitored in the end event (output of the process), and depending on the allocation of the test points, the different activities can be monitored by some of them too. The example shows a  $\#nonIsolatableActivities$  equal to 6, which is the maximum number in the nodes (node WE), corresponding to the

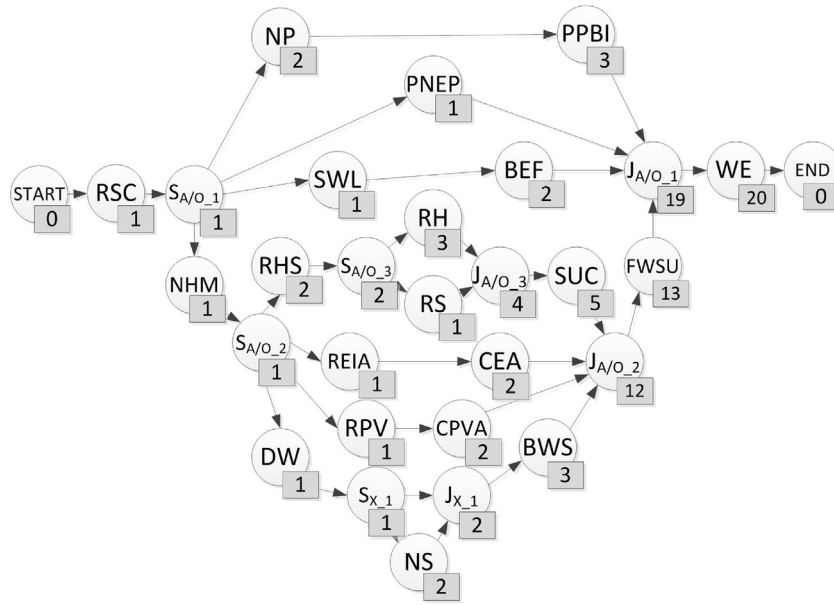


Fig. 9. Example of numbering without any test point allocated.

Table 4  
Test points observations.

Faulty activity	Observations		
	$tp_1$	$tp_2$	$tp_3$
NP, PPBI, PNEP, SWL, BEF or WE	ok	ok	ok
REIA, CEA, RPV, CPVA or FWSU	ok	ok	ab
DW, NS or BWS	ok	ab	ab
RHS, RH, RS or SUC	ab	ok	ab
RSC or NHM	ab	ab	ab

activities NP, PPBI, PNEP, SWL, BEF and WE, which are the largest set of activities that cannot be discriminated between them.

Likewise, Table 4 shows the observations that would be performed by the 3 allocated test points after the detection of an abnormal behaviour of the process (for instance, after a customer complaint). Those observations depend on the activity that

is misbehaving, not existing more than 6 activities whose faulty behaviour produce the same patterns of observable values. For the sake of simplicity, and since the original business process only counts on one XOR split, we suppose that all activities take part in the executed process instances (i.e., for the example, the lower branch of the XOR is executed). Thereby, Table 4 collects the information about the faulty activities and the observations performed by the test points (categorized as normal (ok) or abnormal (ab)).

As a particular example, if a complaint is reported after the execution of a process instance, indicating that the computer for the new employee does not count on the expected software, the observations performed by the test points allow to determine the subset of activities containing the faulty activity. This way, if the test point  $tp_1$  detects the abnormal behaviour, and it is not detected by  $tp_2$ , the problem is caused by the abnormal execution of an activity in the subset {RHS, RH, RS, SUC}, as stated in Table 4. On the other hand, if  $tp_2$  detects the abnormal behaviour too, this is because the

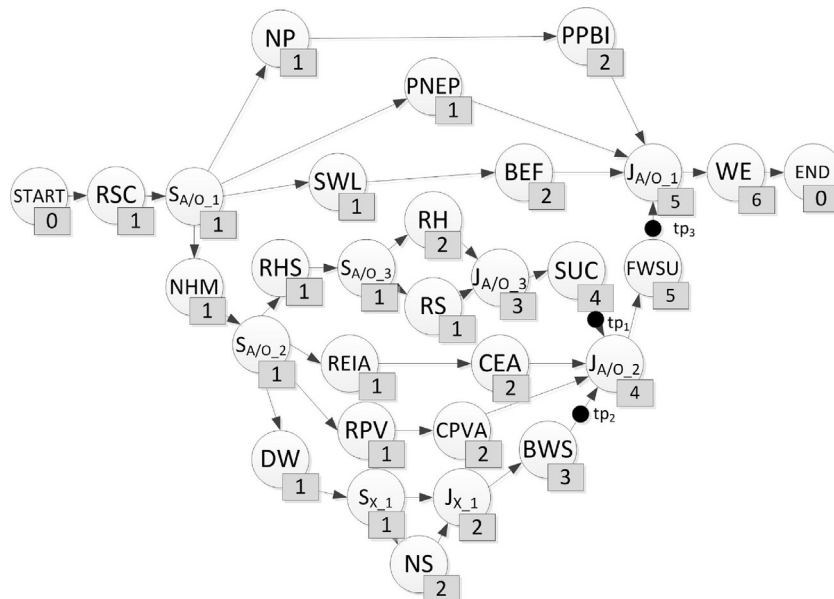


Fig. 10. Example with  $nTestPoints = 3$  for Objective 1.

**Table 5**  
Test points observations.

Faulty activity	Observations				
	$tp_1$	$tp_2$	$tp_3$	$tp_4$	$tp_5$
PPBI, PNEP, SWL, BEF or WE	ok	ok	ok	ok	ok
RH, SUC, REIA, CEA or FWSU	ok	ok	ok	ok	ab
DW, NS or BWS	ok	ok	ok	ab	ab
RPV or CPVA	ok	ok	ab	ok	ab
RHS or RS	ok	ab	ok	ok	ab
NHM	ok	ab	ab	ab	ab
NP	ab	ok	ok	ok	ab
RSC	ab	ab	ab	ab	ab

faulty activity is monitored by both test points  $tp_1$  and  $tp_2$ . Therefore, the activities which are candidates to be faulty are RSC and NHM, as shown in Table 4. This way, the selected test-point allocation enables to determine the source of the abnormal behaviour in the most accurate way for a precise specification of the objective to reach, limiting the checking for the faulty activity to a small subset of activities.

This approach indicates the best test-point allocation within a business process, ensuring a minimal level of diagnosability. Nevertheless, the final validity of the solution depends on the data that can be monitored by the allocated test points (i.e. how they are implemented), and on the later model-based diagnosis process used, such as the techniques presented in Borrego et al. (2009). This is, the minimal diagnosability level obtained can be even reduced in practice if the diagnosis process is able to use the observations from different process instances to exonerate activities, getting a more accurate diagnosis.

### 3.5. Objective 2: minimization of the number of test points to be allocated in order to obtain a fixed diagnosability level

When it is necessary to achieve a determined diagnosability level, a maximum  $\#nonIsolatableActivities$  is fixed, and the objective becomes the minimization of the number of test points to be allocated in order to minimize costs.

In order to model this objective, it is necessary to add more information to the initial CSP. That information is:

- Constraints to limit the  $\#nonIsolatableActivities$  of the business process to a maximum value  $f$ . That value  $f$  indicates the maximum allowed number of non-discriminable activities after the allocation of test points. Therefore, being  $|Act|$  the number of activities in the business process, since  $f = |Act|$  would imply not being necessary to allocate any test point, and  $f = 1$  would imply to allocate test points after every activity, the range of values for  $f$  is defined as  $[2, |Act| - 1]$ .
- The goal to be achieved is given by an objective function which establishes the minimization of the number of test points to be allocated.

#### Constraints

$\#nonIsolatableActivities \leq f$

#### Goal

$minimize(nTestPoints)$

Again for the control flow graph in Fig. 7, and setting a maximum  $\#nonIsolatableActivities \leq 5$ , our approach allocates 5 test points as they are shown in Fig. 11.

Similarly as for Objective 1, Table 5 collects the observations performed by the 5 allocated test points depending on the activity that would be working abnormally.

In this case, there are two sets of 5 activities not discriminable between them:

**Table 6**  
Test cases characteristics.

Number of test cases		100
Size	Min	18
	Max	37
	Avg	27
CFC	Min	5
	Max	12
	Avg	8

- PPBI, PNEP, SWL, BEF and WE, are monitored only at the output.
- RH, SUC, REIA, CEA and FWSU, monitored by  $tp_5$  and at the output.

## 4. Implementation and empirical evaluation

In this section, the implemented tool for the test-point allocation is presented, and the experimental results corresponding to the evaluation performed to it are shown.

### 4.1. Implementation

The Test-Point Allocator tool (Borrego et al., 2012) is implemented, which carries out the allocation of test points in business processes and provides options in accordance with the two objectives detailed in this paper. The tool takes BPMN 2.0 models (OMG, 2011) (generated by using Bonita Open Solution) as input and translates them into CSPs in accordance with the process presented in this paper. As CSP solver, the tool uses the ILOG JSolver™ (JSolver, 2003), which provides a Java library, facilitating its integration with the remainder implemented code. The results are visualized on the screen using the Graphviz/dot library (Graphviz/Dot, 2011). Fig. 12 shows screenshots of the Test-Point Allocator tool: the initial form to upload the input file and choose the options and objective; and the result obtained for one of the tested examples for the process in Fig. 3.

### 4.2. Experimental design

With the aim of performing the evaluation of our tool, our algorithms are executed over a set of randomly generated test cases obtained by using the Process Log Generator (PLG, Burattin and Sperduti, 2010). This mechanism enables the generation of realistic business processes in accordance with certain specific user-defined parameters, such as the determination of the probability of appearance of each type of control flow, or the definition of the maximum number of split-join branches.

In order to take the measurements, 100 test cases have been generated, with different numbers of activities and flows, and using several topologies. Specifically, the characteristics of those test cases are collected in Table 6, which have been measured in accordance two metrics chosen by their direct relation with the diagnosability level of the business processes.

- Size of the business processes, that is equal to the number of activities composing them.
- Control-flow complexity metric (CFC, Cardoso, 2005), which is used to quantify the presence of control nodes in a business process BP, defined as follows:

$$CFC(BP) = \sum_{n \in S_X} CFC_{S_X}(n) + \sum_{n \in S_{A/O}} CFC_{S_{A/O}}(n)$$

where  $CFC_{S_X}(n) = |outedge(n)|$  and  $CFC_{S_{A/O}}(n) = 1$ .

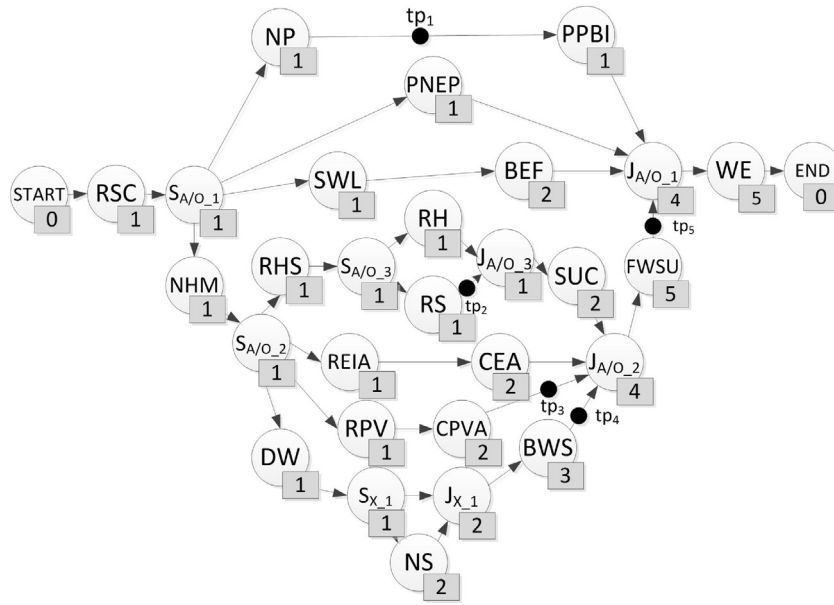


Fig. 11. Example with  $\#nonIsolatableActivities \leq 5$  for Objective 2.

The test cases are measured using a PC with CPU Intel Core 2 Duo 1.86 GHz – 4 GB RAM.

4.3. Experimental results

Since the aim of this contribution is the improvement of the diagnosability of business processes by means of the allocation of test points, the experimental results presented in the following are focused in showing the relation between the diagnosability level and the number of test points allocated for the two objectives discussed in previous sections.

In order to present those results graphically in charts, the business processes used as benchmarks are classified according to the size and CFC metrics.

Fig. 13 shows the results of reaching Objective 1 for both the size of the business processes and their CFC. It is possible to notice that, with the allocation of only a few test points (around 2–3 test points),  $\#nonIsolatableActivities$  scales linearly in a significant way

in both charts, but it tends to remain steady as more test points are allocated.

As regards the experimental results for Objective 2, Fig. 14 shows that the largest diagnosability level is set (i.e., the lower  $\#nonIsolatableActivities$  in the charts), the number of test points also scales linearly with respect to both the size of the business processes and their CFC.

4.4. Performance results

Further to the diagnosability-level analysis, another purpose of the experimental evaluation is to determine the execution time from start to completion of a test-point allocation process. The worst-case complexity for the solution of CSPs is high. Solving CSPs takes exponential time due to the dependency of their complexity on the number of values each variable can take (Dechter, 1992; Traxler, 2008). It is therefore logical to empirically evaluate the performance of the developed methods. In order to assess whether, in

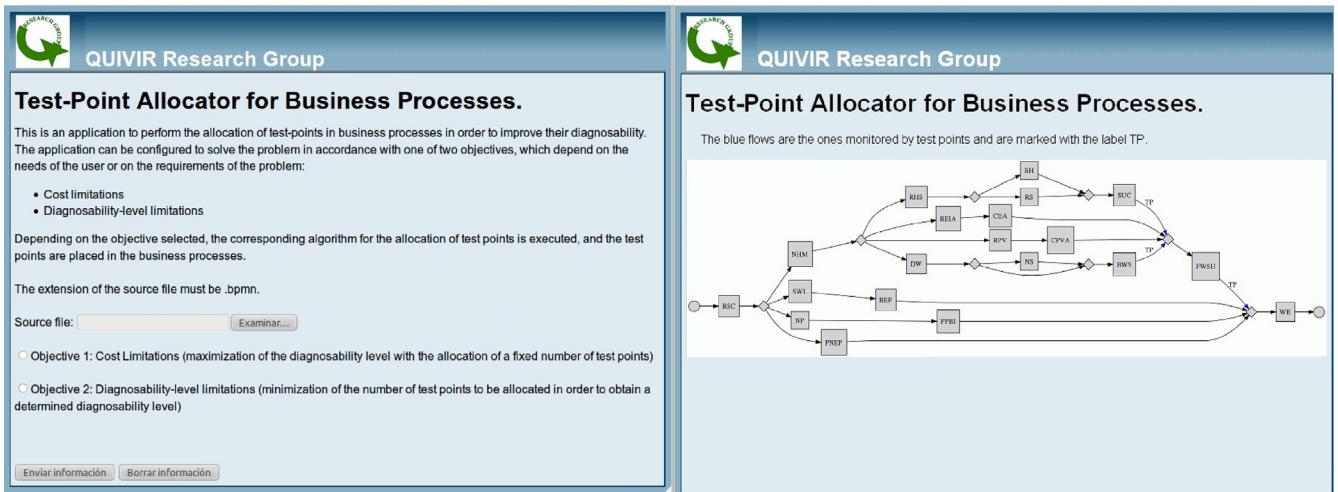


Fig. 12. Screenshots of Test-Point Allocator.

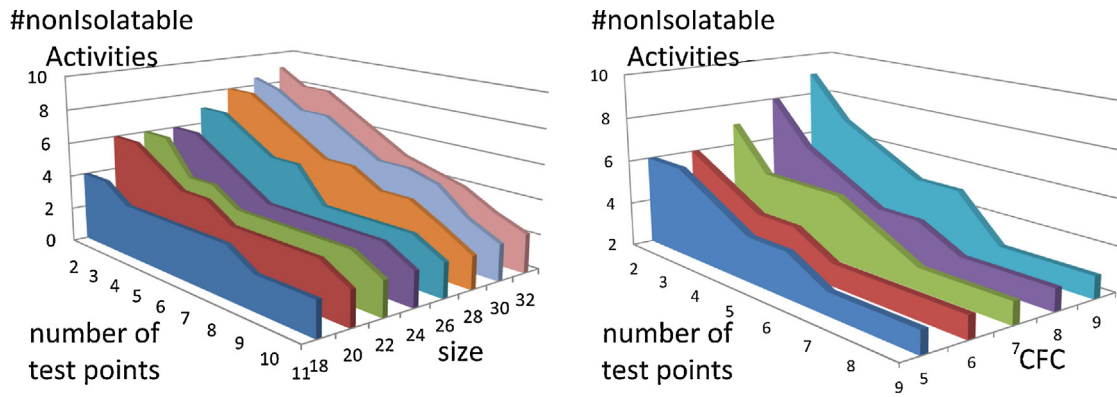


Fig. 13. Experimental results for Objective 1.

practice, the time it takes to allocate test points is acceptable. Hence, in this subsection, the execution times for Objectives 1 and 2 are measured.

Fig. 15 shows the execution time necessary for the complete algorithm to reach Objective 1. In the figure, and for each size of benchmarks, it is possible to notice the increase of the execution time when more test points are allocated. However, it is also possible to observe that the execution time decreases again from a determined number of test point to allocate. This decrease happens because the CSP solver tries every possible combinations of test-point allocations, so that the casuist to check depends on the number of test points to allocate and on the number of flows where those test points can be located (i.e., the size of the process, since the test points can be located in the outgoing flows of the activities). This way, and from a determined quantity of test points to allocate, the number of possible allocations to check decreases, and so do the execution times taken by the CSP solver.

Likewise, the execution time for the complete method for Objective 2 can be observed in Fig. 16. Again, it is possible to see the increase and decrease in the execution time for each size of the benchmarks, even more evident in this case when the value of *#nonIsolatableActivities* is 1.

5. Related work

The diagnosability problem has been analysed in previous work, but never (to the best of our knowledge) it has been adapted to business processes. Moreover, not only is an analysis of the diagnosability required, as performed in Bocconi et al. (2007) and Console et al. (2000), but an improvement of the diagnosability is also necessary.

- Regarding the context of business processes and service workflows, some works in the literature address the diagnosis of behavioural faults (Eshuis and Kumar, 2010; Lin et al., 2002; Van Der Aalst et al., 2011; Zha et al., 2011), focusing on the identification and isolation of modelling errors in workflow designs, at the control flow level, such as deadlocks and lack of synchronization. Likewise, there are previous works dealing with the analysis of system faults (Baresi et al., 2006; Varela-Vaca and Gasca, 2010; Varela-Vaca et al., 2011), which define monitoring rules to detect this kind of fault and activate different remedies. In detail, Varela-Vaca et al. (2011) perform a simulation of security faults over the behaviour of the services (i.e. integrity attacks) in order to diagnose system faults, and Varela-Vaca and Gasca (2010) present an approach based on the allocation of checkpoints. However, these approaches do not perform any automatic determination of monitoring locations.

As far as business faults are concerned, some previous works deal with the problem of fault detection. The approach by Conforti et al. (2011) presents a mechanism to model risks in executable business process models in order to detect them as early as possible during process execution. Nevertheless, as in the case of Alodib and Bordbar (2009), their aim is focused on fault detection, not considering that this mechanism does not guarantee the completeness of a possible later diagnosis.

Previous works such as Yan et al. (2009) and Han et al. (2009) take into account the necessity of monitoring the execution of Web services and business processes in order to count on sufficient information for a later diagnosis process. However, these approaches are focused on the diagnosis, and none of them covers the analysis or improvement of the diagnosability level in business processes.

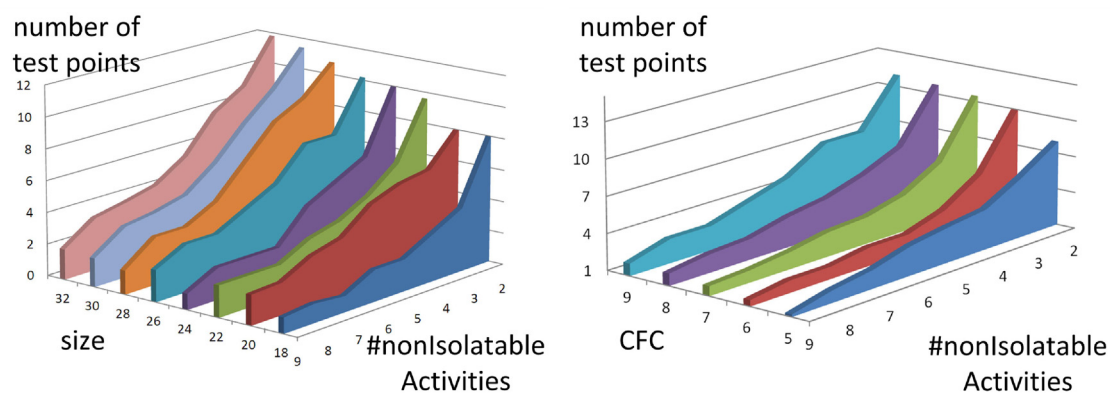


Fig. 14. Experimental results for Objective 2.

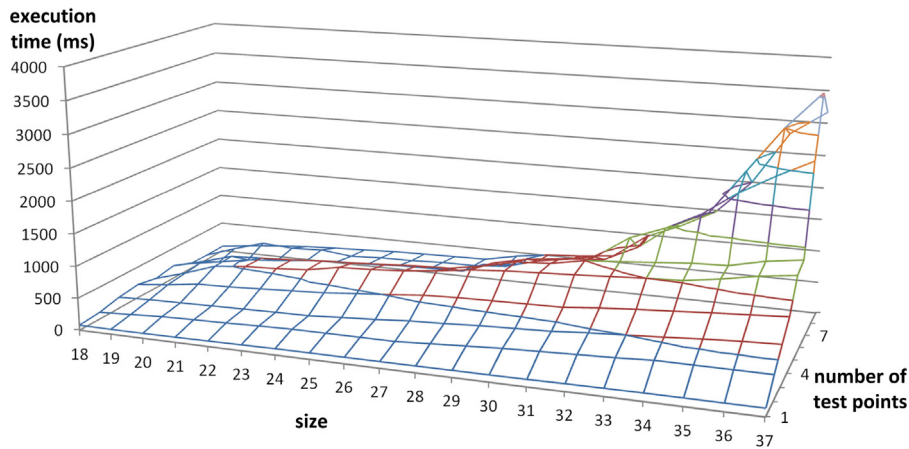


Fig. 15. Execution time for Objective 1.

The contribution presented in Zhang et al. (2009a) presents an algorithm to select locations to perform monitoring, but their selection criteria do not include any diagnosability analysis or cost limitations.

Likewise, although test-point allocation is also studied in the proposal by Narendra et al. (2008), it is not focused on diagnosability either, but on the problem of continuous compliance monitoring at run time in order to prevent non-compliance against policies (security, confidentiality, and data integrity). The selection of locations is useless from the diagnosis point of view since no diagnosability criterion is taken into account.

- With regard to work related in the field of the allocation of sensors in systems composed of many components, a certain number of these systems fail to perform the placement in accordance with the diagnosability (Madron and Veverka, 1992; Maquin et al., 1997; Commault et al., 2006), while other systems allocate the sensors in order to improve diagnosability. A selection of these studies are given in more detail below.

Travé-Massuyès et al. (2006) perform an analysis of many systems in order to study their physical models. This study takes into account the diagnosability criteria by means of a technique that allocates sensors consecutively.

The proposal by Spanache et al. (2004) allocates sensors according to an economic criterion, and presents a method to obtain an optimal-cost sensor system for a certain degree of diagnosability.

Zhang et al. (2009b) allocates test points in circuits and physical systems using a genetic algorithm. Since the signal propagation in this kind of system flows in a different way to that of the dataflow of a business process, this method is inapplicable in the context discussed in this paper.

In the proposal by Ceballos et al. (2005), a new technique is proposed in order to improve the computational complexity for the isolation of faults within a system. The method is based on the addition of new sensors. A CSP (Rossi et al., 2006; Dechter, 2003) is obtained in order to select the necessary sensors to guarantee the problem specification, and an algorithm for the determination of the bottleneck sensors of the system is developed, in order to improve the computational complexity of the CSP.

The contribution presented by Frisk and Krysander (2007) shows an efficient technique that is based on the Dulmage–Mendelsohn decomposition introduced by Dulmage and Mendelsohn (1959). Although the constraint models of most practical systems are under-determined before taking the sensors into account, and over-determined afterwards, this method only applies to just-determined sets of constraints.

A method based on the study of structural matrices (Yassine et al., 2008) proposes that every component of the system is represented by a constraint. Regarding detectability and diagnosability, all subsets of constraints are considered (diagnosable, discriminable, and detectable). The contribution in Yassine et al. (2010) presents a continuation of Yassine et al. (2008), where the Dulmage–Mendelsohn decomposition (Dulmage and Mendelsohn, 1959) together with a combinatorial algorithm are used in the search for the optimal solution.

This paper addresses the topic of allocating test points in business processes in order to improve their diagnosability of business faults. On this topic, to the best of our knowledge, only the proposal by Borrego et al. (2010) has been published, which is a preliminary version of this paper. Furthermore, none of the papers mentioned

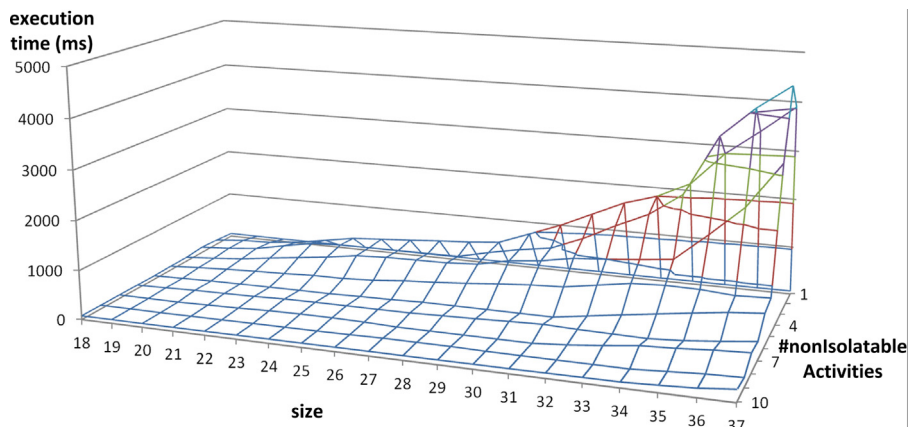


Fig. 16. Execution time for Objective 2.

provide methods to achieve the two objectives sought in this contribution, which considers both the diagnosability level and the cost limitations when it comes to determine the allocation of test points.

## 6. Conclusions and future work

To ensure a minimal diagnosis of a business process in view of abnormal behaviour, the analysis and improvement of its diagnosability is of utmost importance. To this end, we have proposed the allocation of test points to make certain parts of the dataflow observable, thereby facilitating the isolation of the activity or activities which are responsible for possible abnormal behaviour.

The problem of allocating test points in business processes is modelled as Constraint Satisfaction Problems so that it can be efficiently solved using Constraint Programming techniques.

This work presents two different objectives to be achieved, which depend on the requirements of the user or the problem specification. They consider the enhancement of the business processes diagnosability and the cost limitations.

The approach has been implemented in the Test-Point Allocator tool (Borrego et al., 2012). The tool allocates test points in business processes designed using BPMN 2.0, in accordance with the objectives specified.

As for future work, we plan to extend our contribution by considering new features that may condition the distribution of test points along the business processes, such as previous information regarding the fault percentage of each activity, or even runtime characteristics like execution time per activity and the presence of critical or high probability paths.

Likewise, it would be interesting to perform the diagnosis of the business process once the test points are allocated. This diagnosis can provide us with additional information which is useful in the design of a more efficient process for the discovery of the minimal diagnosis. Furthermore, these test points can help to solve the problem of scalability of certain business processes, and may enable an easier diagnosis process to be developed for a business process with numerous activities.

## Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Education (under grant TIN2009-13714), and by FEDER (under the ERDF Program).

## References

- Alodib, M., Bordbar, B., 2009. A model-based approach to fault diagnosis in service oriented architectures. In: Seventh IEEE European Conference on Web Services (ECOWS 2009), Eindhoven, The Netherlands, pp. 129–138.
- Baresi, L., Guinea, S., Plebani, M., 2006. Business process monitoring for dependability. In: WADS, pp. 337–361.
- Bocconi, S., Picardi, C., Pucel, X., Dupré, D.T., Travé-Massuyès, L., 2007. Model-based diagnosability analysis for web services. In: AI'IA'07, pp. 24–35.
- Borrego, D., Gasca, R.M., Gómez-López, M.T., Barba, I., 2009. Choreography analysis for diagnosing faulty activities in business-to-business collaboration. In: 20th International Workshop on Principles of Diagnosis. DX-09, Stockholm, Sweden, pp. 171–178.
- Borrego, D., Gómez-López, M.T., Gasca, R.M., 2012. Test-Point Allocator Tool. <http://www.lsi.us.es/quivir/index.php/Main/Downloads>
- Borrego, D., Gómez-López, M.T., Gasca, R.M., Ceballos, R., 2010. Improving the diagnosability of business process management systems using test points. In: 6th Workshop on Business Process Intelligence (BPI 2010), pp. 194–200.
- BOS, 2011. Bonita Open Solution. BonitaSoft <http://www.bonitasoft.org>
- Burattin, A., Sperduti, A., 2010. Plg: a framework for the generation of business process models and their execution logs. In: Proceedings of the 6th International Workshop on Business Process Intelligence (BPI 2010), September 13. Stevens Institute of Technology, Hoboken, NJ, USA, pp. 214–219.
- Cardoso, J., 2005. Evaluating the process control-flow complexity measure. In: IEEE International Conference on Web Services (ICWS), pp. xxxiii–856.
- Ceballos, R., Cejudo, V., Gasca, R.M., Valle, C.D., 2005. A topological-based method for allocating sensors by using CSP techniques. In: CAEPIA. Springer, pp. 62–68.

- Commault, C., Dion, J.M., Yacoub Agha, S., 2006. Structural analysis for the sensor location problem in fault detection and isolation. In: Proceedings of the IFAC Symposium SAFEPROCESS' 2006, Beijing, China, pp. 2074–2080.
- Conforti, R., Fortino, G., Rosa, M.L., ter Hofstede, A.H., 2011. History-aware, real-time risk detection in business processes. In: Proceedings of the 19th International Conference on Cooperative Information Systems (CoopIS), pp. 100–118.
- Console, L., Picardi, C., Ribaud, M., 2000. Diagnosis and diagnosability analysis using process algebra. In: Proc. 11th Int. Workshop on Principles of Diagnosis (DX), pp. 25–32.
- Dechter, R., 1992. Constraint Networks (Survey). In: Encyclopedia of Artificial Intelligence, second ed. John Wiley & Sons, Inc., pp. 276–285.
- Dechter, R., 2003. Constraint processing. Elsevier Morgan Kaufmann, ISBN 978-1-55860-890-0, pp. 1–XX, 1–481.
- Dressler, O., Struss, P., 2003. A toolbox integrating model-based diagnosability analysis and automated generation of diagnostics. In: The 14th International Workshop on Principles of Diagnosis (DX03), Washington, DC, USA, pp. 99–104.
- Dulmage, A.L., Mendelsohn, N.S., 1959. A structure theory of bi-partite graphs of finite exterior extension. Transactions of the Royal Society of Canada 53, 1–13.
- Eshuis, R., Kumar, A., 2010. An integer programming based approach for verification and diagnosis of workflows. Data & Knowledge Engineering 69, 816–835.
- Frisk, E., Krysander, M., 2007. Sensor placement for maximum fault isolability. In: Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07), pp. 106–113.
- Graphviz/Dot, 2011. <http://www.graphviz.org>
- Han, X., Shi, Z., Niu, W., Lin, F., Zhang, D., 2009. An approach for diagnosing unexpected faults in web service flows. In: Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing, pp. 61–66.
- JSolver 2.1, 2003. Reference manual (unpublished).
- Lanz, A., Weber, B., Reichert, M., 2012. Time patterns for process-aware information systems. Requirements Engineering, 1–29.
- Lin, H., Zhao, Z., Li, H., Chen, Z., 2002. A novel graph reduction algorithm to identify structural conflicts. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), vol. 9. IEEE Computer Society, Washington, DC, USA, p. 289.
- Madron, F., Veverka, V., 1992. Optimal selection of measuring points in complex plants by linear models. AIChE Journal 38 (2), 227–236.
- Maquin, D., Luong, M., Ragot, J., 1997. Fault detection and isolation and sensor network design. European Journal of Automation 31, 393–406.
- Narendra, N., Varshney, V., Nagar, S., Vasa, M., Bhamidipaty, A., 2008. Optimal control point selection for continuous business process compliance monitoring. In: IEEE International Conference on Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. IEEE, pp. 2536–2541.
- OMG, 2011. Object Management Group, Business Process Model and Notation (BPMN) Version 2.0. OMG Standard, Available at: <http://www.bpmn.org/>.
- Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. Elsevier Science Inc., New York, NY, USA, ISBN 0444527265.
- Spanache, S., Escobet, T., Travé-Massuyès, L., 2004. Sensor placement optimisation using genetic algorithms. In: Proceedings of the 15th International Workshop on Principles of Diagnosis, DX-04, pp. 179–183.
- Travé-Massuyès, L., Escobet, T., Olive, X., 2006. Diagnosability analysis based on component-supported analytical redundancy relations. IEEE Transactions on Systems, Man, and Cybernetics, Part A 36, 1146–1160.
- Traxler, P., 2008. The time complexity of constraint satisfaction. In: Proceedings of the 3rd International Conference on Parameterized and Exact Computation. Springer-Verlag, Berlin, Heidelberg, pp. 190–201.
- Van Der Aalst, W., Van Hee, K., Ter Hofstede, A., Sidorova, N., Verbeek, H., Voorhoeve, M., Wynn, M., 2011. Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects of Computing 23, 333–363, Cited by (since 1996) 20.
- Varela-Vaca, A.J., Gasca, R.M., 2010. Opubs: fault tolerance against integrity attacks in business processes. In: 3rd International Conference on Computational Intelligence in Security for Information Systems (CISIS'10), pp. 213–222.
- Varela-Vaca, A.J., Gasca, R.M., Borrego, D., Pozo, S., 2011. Fault tolerance framework using model-based diagnosis: towards dependable business processes. International Journal on Advances in Security 4, 11–22.
- Weske, M., 2007. Business Process Management. Concepts, Languages, Architectures. Springer-Verlag New York, Inc., Secaucus, NJ, USA, ISBN 3540735216.
- Yan, Y., Dague, P., Pencolé, Y., Cordier, M.O., 2009. A model-based approach for diagnosing faults in web service processes. International Journal of Web Services Research JWRS, 87–110.
- Yassine, A.A., Ploix, S., Flaus, J.M., 2008. A method for sensor placement taking into account diagnosability criteria. Applied Mathematics and Computer Science 18, 497–512.
- Yassine, A.A., Rosich, A., Ploix, S., 2010. An optimal sensor placement algorithm taking into account diagnosability specifications. In: International Conference on Automation, Quality and Testing, Robotics, vol. 2, pp. 1–6.
- Zha, H., van der Aalst, W., Wang, J., Wen, L., Sun, J., 2011. Verifying workflow processes: a transformation-based approach. Software and Systems Modeling 10, 253–264.
- Zhang, J., Chang, Y., Lin, K.J., 2009a. A dependency matrix based framework for QOS diagnosis in SOA. In: IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2009, Taipei, Taiwan, pp. 1–8.
- Zhang, Y., Chen, X., Liu, G., Qiu, J., Yang, S., 2009b. Optimal test points selection based on multi-objective genetic algorithm. In: IEEE Circuits and Systems International Conference on Testing and Diagnosis, 2009. ICTD 2009. IEEE, pp. 1–4.



**Diana Borrego** is a lecturer in Computer Science at the University of Seville, Spain. She received the MSc degree (2006) and PhD degree (2012) in Computer Science from the University of Seville. Diana's research is focused on the application of Artificial Intelligence techniques for the quality improvement of business process management systems. Specifically, her research interests include constraint programming, fault diagnosis and business process management. Diana Borrego has published papers at international conferences and journals. She is a member of the Quivir Research Group at the University of Seville. She can be contacted at dianabn@us.es.

**María Teresa Gómez-López** received her degree in Computer Science from the University of Seville in 2001 and her PhD degree in 2007. She is currently a PhD Lecturer in the Department of Languages and Computer Systems at the

University of Seville, Spain. She worked as a researcher in the Spanish National Research Council. She has participated in Spanish research projects, and technology transfer projects, publishing numerous articles in Computer Science journals and international conferences. Currently, her main research interests are Business Compliance Rules, Business process management and Constraint Databases.

**Rafael M. Gasca** received his PhD degree in Computer Science in 1998. He is an Associate Professor at the University of Seville (Spain) where he has been responsible for the Quivir Research Group since 1999. His current research is focused on fault diagnosis and security technologies for business process management systems. He has been involved in European and Spanish research projects and has published numerous articles in Computer Science journals and international conferences.