Contract-based Diagnosis for Business Process Instances using Business Compliance Rules

D. Borrego, R. M. Gasca, M. T. Gómez-López, L. Parody

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Sevilla, Spain
{*dianabn, gasca, maytegomez, lparody*}*@us.es*

ABSTRACT

In order to increase the quality of business processes when they are automated, the correctness of the activities can be checked by means of an analysis of the corresponding business compliance rules. By analyzing the trace of an instance of a business process, it is possible to detect the correctness of the process and to determine which activity is faulty. Each activity or set of activities is related to a set of business compliance rules, which work as contracts that the activities must satisfy throughout the dataflow.

In order to diagnose a business process instance, not all the activities participate in every single execution, since there are control flows that permit the execution of several branches for a varied number of times. We propose to automate the diagnosis of these executions of a business process taking into account the involved activities and their business compliance rules. Our main contributions are related to the construction of the corresponding framework using several techniques related to the constraint programming paradigm to obtain the incorrect activities. The two different proposals consider the tradeoff between the obtaining of the minimal diagnosis and the performance.

Key words: Model-based diagnosis, Constraint programming, Business rules, Minimal unsatisfiable subset

1 INTRODUCTION

A business process consists of a set of activities that are performed in coordination in an organizational and technical environment (Weske, 2007). Business Process Management (BPM) includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of a business process. The base of BPM is the explicit representation of business processes with their activities and the contract between them. In order to describe the workflow of a process, it is possible to use control flow operators, such as *and*, *xor*, *or*, *loops*... The execution of these operators can be determined by logical conditions of data which define the instance executed for each example. Likewise, it is also possible to describe the behavior/semantics of the data managed in the business process by using compliance rules. Business rules can also be used to describe the contract that each activity has to satisfy for a whole process. The rules that describe the company policies associated to an activity or set of activities are called Business compliance rules. Since each organization defines its own policies about things such as prices, costs, employee numbers, deadlines of tasks and so on, it is possible to specify a different set of rules for each business process, even for different activities in the same business process.

We consider that the business process contract consists of the following:

- The organizational model which consists of organizational units and associated roles;
- The data model that includes the business compliance rules for these data;
- A business process model which in our case is a BPMN diagram. BPMN is a modelling language, chosen for the following reasons: (a) it is the emerging standard for business modelling and (b) it has a transformation into executable code and platforms, which can then be used for automatic processing of compliance rules.

The activities transform the data of the business objects, therefore it is necessary to check whether the compliance rules are satisfiable after data transformation. In the negative case, those activities responsible for the non-compliance are identified. A declarative perspective of the objectives of processes, which indicates what needs to be done in order to comply with the constraints, is available.

In the previous work (Guillou *et al.*, 2009), it is proposed a diagnosis based on the propagation between different local diagnoses. This is an interesting proposal but it is not always possible to equip each service with a local diagnosis and to work with chronicles. For

these reasons, we propose a solution based on to describe the tasks' behaviour with a contract represented by constraints.

Extensive literature research regarding business process compliance has been presented (Sadiq *et al.*, 2007) (Namiri and Stojanovic, 2007) (Ghose and Koliadis, 2007). However, the main question arising in this article is whether it is possible to infer what or where a failure has occurred with respect to the business policies and procedures in terms of the dataflow values. This suggests an adaptation of the typical fault model-based diagnosis approach to business process diagnosis, which is what is pursued in this work. For the contract/compliance management domain, compliance auditors are necessary. For the BPM we propose a framework that detects, in a run-time way, the non-compliance of the rules, performing an automatic compliance diagnosis.

In order to automate the diagnosis process, we will represent as constraints the business compliance rules associated to the different activities. These constraints are linear or polynomial equations or inequations over dataflow variables, related by a boolean combination (*and/or*), whose float or integer variables are defined in a domain.

Current business rules could be integrity rules, derivation rules, reaction rules, production rules and transformation rules. This work focuses on the production rules, whose representation is the *if Condition then Consequence* format, and are used to validate the contract of the business process, where *Condition* represents the constraint to validate and *Consequence* the evaluation of the compliance rule. By using *Constraints* the information is represented at a more abstract level, since languages based on constraints include and improve all the capacity of representation of current rules engines, such us Drools, Fair Isaac Blaze Advisor, ILOG JRules and Jess. Likewise, representing the business compliance rules as constraints we can get the automatization of the contract-based diagnosis process.

By using business compliance rules and artificial intelligence techniques oriented towards diagnosis, it is possible to describe the contract of the process to validate the correctness and determine any incorrect activities. In the business process area, the diagnosis process has two condition that render the task of diagnosis more complicated: (a) Depending on the process instance and the input data, certain activities will work towards obtaining the objective of the process. This implies that in the diagnosis process, it is necessary to know which activities have participated. (b) A branch loop is a possible scenario, where a set of activities are executed several times, thereby the different executions of the loop have to be taken into account.

For solving the previous problems, two different strategies are proposed. In order to present them, the structure of the paper is as follows. Section 2 contains the notation and formal definitions necessary to clarify our proposal. In Section 3 we present the specification of a motivating example of business processes and business compliance rules that is used in a diagnosis process. A framework to automate the fault diagnosis is presented in Section 4. Section 5 contains the main contribution of the work, an improved algorithm based on minimal unsatisfiable subsets to find any incorrect activities by taking into account the executed activities and the possible loops. Section 6 presents the experimental results, and Section 7 concludes the paper by discussing related works and pointing out future lines of research.

2 NOTATION AND FORMAL DEFINITIONS

The purpose of this section is to give a concise formal development of the basic notions of business process theory in order to clarify the concepts necessary in the later sections.

Definition 1. Business compliance constraint. A representation of a business compliance rule as a constraint. There are two kinds:

- Global compliance constraints: constraints that involve all the activities of the business process.
- Activity compliance constraints: constraints that involve only some activities of the business process.

A Constraint Satisfaction Problem *CSP* is built from the business compliance constraints and the variables involved on them, for two main reasons: *(i)* the important number of efficient solvers for *CSPs* that exist, and *(ii)* in order to automate the diagnosis process.

A *CSP* represents a reasoning framework consisting of variables, domains and constraints. Formally, it is defined as a triple <*X*, *D*, *C*> where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \ldots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \ldots, C_m\}$ is a set of constraints. A constraint $C_i = (V_i, R_i)$ specifies the possible values of the variables in V simultaneously in order to satisfy R.

In the built *CSP*, the variables' domains depend on either the defined contract or the specification provided by the role of the pool.

The different business compliance constraints are related to the different activities of the business process. That relation can be performed in an automatic way, since the data, transformed by the activities, participate in the constraints as variables.

The Compliance Constraint/Activity Mapping is an association such that any business compliance constraint is related to a set of activities of a business process.

In order to automate the computation of the minimal diagnosis, it is necessary to define the following concepts:

A Reified Constraint is a business compliance constraint that is associated to truth values which represent the associated activities according to the above mapping.

Definition 2. Minimal Unsatisfiable Subset (*MUS*). Given a set of business compliance constraints C, a *MUS* of C is a subset of C that is (1) unsatisfiable and (2) minimal, in the sense that removing any one of its elements makes the rest of the *MUS* satisfiable. That is, m is a *MUS* of C iif $m \subseteq C : m$ is unsatisfiable, and $\forall c \in m, m \backslash \{c\}$ is satisfiable.

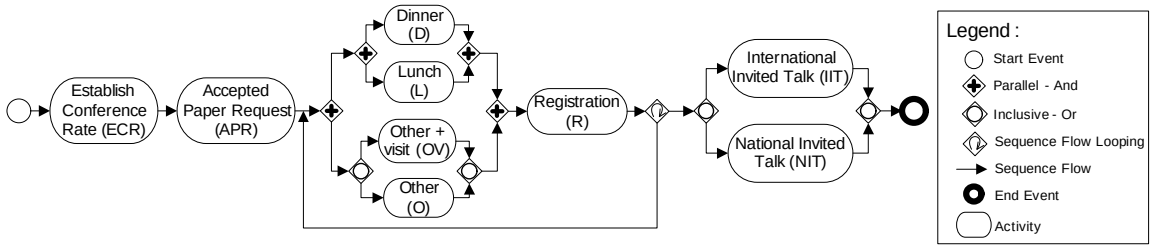

Figure 1: Motivating Example

3 MOTIVATING EXAMPLE

In the following, a partial example of a business process for the handling of a conference for an organization is described. The process starts with the Call for Papers. The technical secretary receives the inscriptions of the participants. The various components of the organizing committee then decide on the meals and material cost in order to obtain the final approval, which is carried out by the organizing committee president. Upon all approvals the conference begins.

Figure 1 depicts an example process graph in standard BPMN notation. In fact, this example is based on a BPMN diagram example from the BPMN 1.1 specification (OMG 2008).

The process of Figure 1 has three types of control flow operators (*or split/join*, *and split/join* and *sequence flow looping*). These control flow operators enable several activities to be performed in terms of the number of accepted papers, the quantity of money offered by the partners, the capacity of the restaurants and the number of participants.

The variables with their respective domains and the business compliance constraints that will be used to validate and diagnose each instance of the process are shown in Figure 2.

The variables are: $totalG$ as the total cost of the conference; $totalI$ as the total income; $NumPT$ as the number of assistants; $PartC$ as the registration fee; $NumAccPaper$ as the number of accepted papers; $partnerPart$ as the income on behalf of the partners; $oParC$ as other costs; $lunchC$ as the cost of each lunch; $dinnerC$ as the cost of the gala dinner; $gastosP$ as the total expenses for each participant; and $invitedTalk$ as the expenses incurred by the invited speakers.

In order to organize the conference some activities are executed, but some of them may be incorrect according to the main goals of the conference committee: ”Do not spend more money than obtained with registrations and partner incomes; and a high participant satisfaction degree with the conference organization”.

It is possible that, after an execution of the process, and due to the values assigned to some variables of the *CSP* (decisions taken by the organizing committee), the business compliance constraints are not satisfiable. In this case a diagnosis process must be executed to find out which activity or activities are not correct.

Variables:

totalG, D : {25000..70000}
totalI, D : {25000..70000}
NumPT, D : {75..170}
PartC, D : {200..390}
NumAccPaper, D : {50..80}
partnerPart, D : {4000..15000}
oParC, D : {30..185}
lunchC, D : {10..30}
dinnerC, D : {60..100}
gastosP, D : {0..70000}
invitedTalk, D : {0..10000}

Compliance Constraints:

totalG <= totalI
*totalG >= totalI*0.8*
*totalG==NumPT*gastosP+ InvitedTalk*
*totalI==NumPT*PartC+partnerPart*
*PartC*0.10<=dinnerC*
*dinnerC <= PartC*0.35*
*PartC*0.10 <= 3*lunchC*
*3*lunchC<=PartC*0.35*
*NumPT.getMin()<=75 => oParC <= 0.20*PartC + 0.05*partnerPart*
*NumPT.getMin()<=75 => oParC >= 0.05*PartC + 0.05*partnerPart*
*NumPT.getMin()>75 => oParC<=0.25*PartC*
*NumPT.getMin()>75 => oParC>=0.05*PartC*
*gastosP == 3*lunchC + dinnerC + oParC*
*NumAccPaper*1.8 >= NumPT*
*NumAccPaper*0.5 <= NumPT*
*partnerPart<=2000 => InvitedTalk >= 0.2*partnerPart*
*partnerPart<=2000 => InvitedTalk<= partnerPart+0.10*PartC*NumPT*
*partnerPart>2000 => InvitedTalk >= 0.4*partnerPart*
partnerPart>2000 => InvitedTalk <= partnerPart

Figure 2: Example of business compliance constraints

4 FRAMEWORK FOR THE CONTRACT-BASED DIAGNOSIS OF BUSINESS PROCESSES

In classic model-based diagnosis, in order to determine which component is working incorrectly, it is neces-

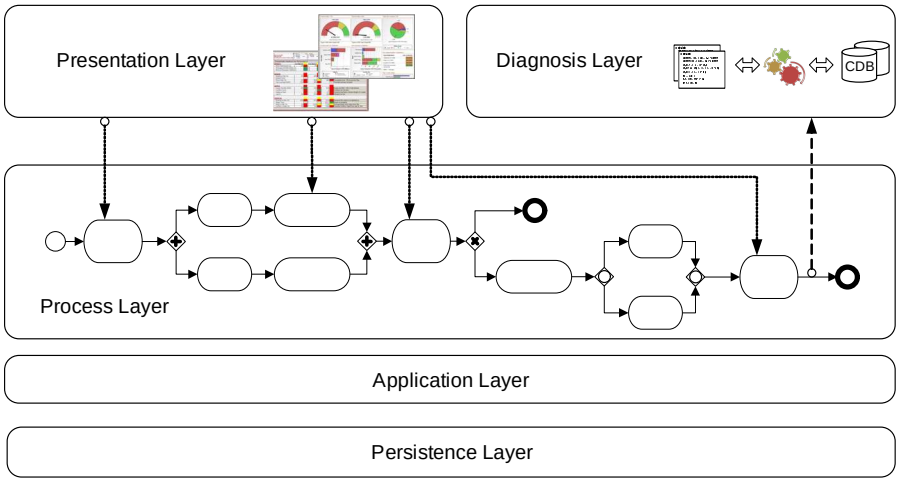

Figure 3: Proposed Framework

sary to compare the model of the system with the observable variables. In the business process area, details of the model may be unavailable due to company privacy, but the policy of the companies and what constraints have to be satisfied by the dataflow must be known. This allows the user to diagnose possible faulty activities and to understand why a particular instance of the business process is not working. However, determining the root cause of a failure is an even greater challenge, since the topology of the business process and the involved activities for each instance must all be taken into account.

Along these lines, we propose a framework based on the analysis of business output data and business compliance rules to diagnose a business process instance. The model of the business process and the diagnosis is based on the business compliance constraints.

The proposed framework is an extension of the classic Process Aware Information System (PAIS) framework. In general, PAIS architecture can be viewed as the 4-tier system as presented in (Weber *et al.*, 2009), where from top to bottom the layers are: Presentation Layer, Process Layer, Application Layer and Persistency Layer. As a fundamental characteristic, PAIS provides the means to separate process logic from application code. At runtime the PAIS then orchestrates the processes according to the defined logic and coordinates the processing of relevant applications and other resources. The new framework, shown in Figure 3, presents a new layer where the diagnosis of the business process is performed. Likewise, the Presentation Layer counts on a Workflow Dashboard in order to permit the participation of the role of the pool at runtime in an equivalent manner to the way that the classic model-based diagnosis uses the observational model for the sensors.

The Diagnosis Layer for the business processes is a parallel and ”independent” system from the process layer. They are independent since it can be executed in different machines, by different actors, and at the same time.

Figure 3 gives an overview of our framework. Processes are modelled in terms of a typical workflow language, featuring task nodes (the activities carried out inside the process) as well as parallel splits/joins and xor splits/joins to model the control flow. Such a model specifies only which sequences of activities, i.e. which execution paths, may occur; it cannot model more subtle or indirect dependencies between the activities, due to a compliance rule which expresses an obligation in that, activity A must always be checked for compliance with the process iff B's precondition is always guaranteed to be true.

The business compliance rules are checked against the logical states that can be traversed by the process. A naive way of checking compliance is hence to enumerate all those states. Clearly, given that the number of states is (in general) exponential to the size of the process, such an approach is undesirable. A human modeller will not tolerate long waiting times during process modelling, and checking the compliance of a whole process repository against an altered constraints base may become completely infeasible if every single process involves a state enumeration. The question hence is: do restricted cases exist where we can check compliance efficiently? And can we devise approximation techniques for more general cases? Regarding loops, as stated, the original definition of basic processes (Weber *et al.*, 2009) disallows them.

The new layer added to the framework is detailed below.

4.1 Diagnosis Layer

When a business process model using business compliance constraints is analyzed, some inconsistencies can be identified. In order to determine these inconsistencies, three types of information about the business process instance are necessary:

- Business compliance rules related to the business process instance.
- The activities that have participated in the process instance obtained from a log file, since the various conditions associated to the control flow operators can determine separate execution paths.

- The values of the variables of dataflow that participate in the business compliance rules. The variables of dataflow are those which can be instantiated during the execution of the process.

With this information, the diagnosis layer has to execute three steps:

1. Select which business compliance constraints are related to the business process in accordance with the log file (Compliance Constraint/Activity Mapping).
2. Detect whether the business process has worked correctly in this instance. This step is carried out by building a Constraint Satisfaction Problem (CSP) with the involved business compliance constraints and the dataflow instance.
3. If an inconsistency is detected, the diagnosis process is used to determine the activity or activities which are responsible of the non-compliance.

In order to perform the aforementioned determination of inconsistences and the later diagnosis, the business compliance rules and the dataflow are mapped into a *CSP*. This way, the diagnosis task can be performed by using two proposals of diagnosis techniques, as explained in Section 5.

The Diagnosis Layer has to store all the business compliance rules and to obtain them in an efficient way to determine which business compliance rules are involved in the diagnosis process. As the business compliance rules are represented by constraints, Constraint Databases (*CDBs*) can be used to support and handle these rules.

When dealing with a great quantity of data, the use of a database is a mandatory decision. The storage of business rules also implies storing all the details related to their variables, the domain of variables and data persistence relationships. These types of information and business rules expressed by Constraints are supported by Constraint Database Management Systems (CDBMS).

CDBs were initially developed in 1992 (Kanellakis *et al.*, 1992). The basic idea behind the CDB model is to generalize the notion of a tuple in a relational database to a conjunction of constraints, since a tuple in relational algebra can be represented as an equality constraint between an attribute of the database and a constant. In a real business process, great quantity of compliance rules must be defined, hence a repository is required in order to evaluate these rules as soon as possible. The CDB used in this paper is based on Labelled Object-Relational Constraint Database Architecture (LORCDB Architecture) (Gómez-López *et al.*, 2009).

5 CONTRACT-BASED DIAGNOSIS OF BUSINESS PROCESSES

The previous section presents a proposed framework composed of several layers. The Diagnosis Layer, added to attain the objective of this paper, deploys two strategies in order to provide two different diagnosis solutions, both of which are based on the Constraint Programming paradigm. Due to the different control flow patterns that form the structure of a business process, it is necessary to take the instance that has been executed into account in order to determine the constraints that will compose the final *CSP* to be solved, in the same way as for the input data introduced during the execution of the process. That is, the information about the input data and the activities that have been executed are of interest since they will define which business compliance constraints are related to that instance and whether these constraints are satisfiable for the input data in that instance of the process.

Therefore, the Diagnosis Layer of the framework receives the activities that have been executed in order to build the correct *CSP*. The control flow patterns that influence that execution trace are:

- **Parallel Split (AND).** This is a single thread of execution that splits into two or more parallel branches which execute activities concurrently. Since all branches are executed, if the execution of the business process reaches this structure, then all the activities, and consequently all their related business compliance constraints, will be taken into account in the diagnosis process.
- **Multi-choice (OR/XOR).** This is a single branch that diverges into two or more branches. When the execution of the single branch has finished, the thread of control is passed to one or more branches depending on the evaluation of a logical expression. Therefore, the *CSP* to solve will be composed of only those business compliance constraints associated to the activities in the executed branches.
- **Loop.** This structure enables a group of activities of the business process to be executed at least once depending on a logical condition. With regard to the incorporation of business compliance constraints into our *CSP*, and depending on the kind of tasks performed by the activities within the loop, two different cases can come up: (1) when only the most recent execution of the activities in the loop must be considered, since previous executions only performed tasks that are cancelled or deleted by the final execution; (2) when all the executions of the loop must be taken into account in the building of the *CSP*, since all the executions of the activities perform relevant actions with the input data.

  In the first case, the business compliance constraints and input data incorporated to the *CSP* are those related to the activities executed in the last iteration of the loop.

  In the second case, the constraints derived from the business compliance rules of the activities in the loop must be repeated in the *CSP* as many times as they are executed. In order to differentiate between the input data introduced in each iteration of the loop, the variables instantiated though the input data are renamed so that they have different names for each iteration of the loop, both as variables in the *CSP* and as variables in the different repeated constraints.

Once the constraints of the *CSP* are determined, based on the execution trace of the business process, then the diagnosis can be performed with either of these two proposed solutions:

- **Diagnosis using reified constraints and MaxCSP.** This proposal takes the relation between each business compliance constraint and one or more activities of the business process into account. Each business compliance constraint is transformed into a reified constraint, giving rise to an overconstrained *Constraint Satisfaction Problem*. Due to not existing any solution, there is an error. Using the reified constraints to build a MaxCSP, we try to minimize the number of unsatisfied constraints, finding the minimal sets of faulty activities by means of the calculation of the minimal hitting sets.
- **Diagnosis based on the attainment of the MUSes.** The idea of this proposal is to find the Minimal Unsatisfiable Subsets of constraints (*MUSes*, (de la Banda *et al.*, 2003)) in order to find what is wrong in an overconstrained *CSP* instance. By finding these subsets, and calculating their minimal hitting sets, the activities responsible for rendering the business compliance constraints of the business process inconsistent can be identified (Reiter, 1987). This proposal obtains the minimal diagnosis.

Both proposals for diagnosis are given in greater detail in the following subsections.

5.1 Diagnosis Using Reified Constraints and MaxCSP

As mentioned above, each business compliance constraint is associated to the activities related to the participating variables. This association is performed in an automatic way, since it is only necessary to determine the variables related to each activity in order to be able to relate business compliance constraints and activities.

This proposal builds reified constraints based on each business compliance constraint. It is necessary to add new variables to our *CSP* to be solved. They are boolean variables, one for each business compliance constraint, which are used to build reified constraints, associating each constraint to a boolean through an equality.

Applying this idea to the example in Figure 1 gives rise to 19 new boolean variables, one for each constraint ($C1, C2, C3, \ldots$). The compliance constraints are related to these boolean variables, forming reified constraints as, for example:

$$C1 == (totalG <= totalI)$$
$$C2 == (totalG >= totalI * 0.8)$$

Finally, the objective function is added to the *MaxCSP* to solve. Its aim is to maximize the number of activities which are not responsible of the unsatisfactibility of the CSP. That is, the objective function maximizes the number of boolean variables instantiated to *true*.

Once this maximization is performed, it is necessary to know the activities responsible for the inconsistency. In order to do that, we take the sets of activities related to the constraints whose associated boolean has been instantiated to false, and find out the minimal collection of activities which cover all those sets. According to diagnosis theory, the best way to find that activity or activities is by calculating the hitting sets and minimal hitting sets, whose definition is as follows:

Definition 3. Hitting Set (HS) for a collection of sets of components $\mathcal{C}$ is a set of components $\mathcal{H} \subseteq \bigcup_{\mathcal{S} \in \mathcal{C}} \mathcal{S}$ such that $\mathcal{H}$ contains at least one element for each $\mathcal{S} \in \mathcal{C}$.

Definition 4. Minimal HS. An HS of $\mathcal{C}$ is minimal iff no proper subset of $\mathcal{H}$ is an HS of $\mathcal{C}$. The minimal HSs for a set of sets are formed by $\{\mathcal{H}_1, \mathcal{H}_2, \ldots \mathcal{H}_n\}$, where $\mathcal{H}_i$ is a minimal HS of components. The cardinality of $\mathcal{H}_i$ ($|\mathcal{H}_i|$) is the number of components of $\mathcal{H}_i$.

The calculation of these minimal hitting sets of activities provides us with the minimal diagnosis for a business process instance.

As an example, for the reified constraints of the motivating example in Section 3, if the organizing committee of the conference receives less money that expected from the partners, and the *CSP* is solved, the activities instantiated to false are *ECR* and *IIT*. It means that the lack of money must be solved increasing the conference rate, or decreasing the expenses of the international invited talk.

5.2 Determination of MUSes for Contract-based Diagnosis

This implementation is based on the attainment of all the *MUSes*, which represent the most succinct explanations, in terms of the number of involved constraints, of infeasibility. Indeed, when we check the consistency of a *CSP*, it is preferable to know which constraints are contradicting one another rather than only knowing that the *CSP* as a whole is inconsistent.

Due to the computational complexity of the attainment of all *MUSes*, some existing approaches were designed to derive only some of the *MUSes* and not all of the *MUSes* of an overconstrained *CSP*. Our proposal is based on an improvement in (Gasca *et al.*, 2007), which is grounded in structural analysis in order to determine all the *MUSes* efficiently. In that paper, several techniques are presented, which improve the complete technique in several ways depending on the structure of the constraint network. These techniques make use of the powerful concept of the structural lattice of the constraints and neighbourhood-based structural analysis to boost the efficiency of the exhaustive algorithms. Since systematic methods for solving hard combinatorial problems are too computational expensive, structural analysis offers an alternative approach for fast generation of all *MUSes*. Accordingly, experimental studies of these new techniques outperform the best exhaustive techniques since they avoid the necessity of solving a high number of *CSPs* with exponential complexity. However they do add certain new procedures with polynomial complexity.

In the case of this paper, two techniques are applied so that their computational complexity can be compared:

- **Exhaustive technique.** It attains the *MUSes* by means of a queue and a list to store the subsets of satisfied and unsatisfied business compliance constraints, respectively (Algorithm 1).
- **Variable-Based Neighbourhood technique.** It uses the knowledge about the Variable-Based

Algorithm 1 Algorithm to Obtain the *MUSes* (I)

//Being C the compliance constraints that presents the inconsistency:
$Q :=$ Queue initialized with each constraint in C
$MUS :=$ List that will contain the *MUSes*, initialized to empty
while Q is not empty **do**
  $\{c_i \dots c_j\} := Q.poll()$ //retry and remove the head of Q
  for $c_k \in \{c_{j+1} to c_n\}$ **do**
    if NOT $\exists SubSet^{1\dots n-1}_{\{c_i \dots c_j\}} \cup c_k \in MUS$ //n is cardinality of $\{c_i \dots c_j\}$ **then**
      if $\{c_i \dots c_j\} \cup c_k$ is satisfiable **then**
        $Q.add(\{c_i \dots c_j\} \cup c_k)$
      else
        $MUS.add(\{c_i \dots c_j\} \cup c_k)$
      end if
    end if
  end for
end while

Neighbourhood explained in (Gasca *et al.*, 2007). To this end, this technique relates business compliance constraints as neighbours if they share certain *Non-Observable Variables*, which are those variables whose values remain unknown until the end of the execution of the process since they are not determined by the role of the pool.

Algorithm 2 shows the details of this process.

Algorithm 2 Algorithm to Obtain the *MUSes* (II)

//Being C the compliance constraints that presents the inconsistency:
$Q :=$ Queue/Stack initialized with each constraint in C. //The data structure selected determines the search strategy.
$MUS :=$ List that will contain the *MUSes*, initialized to empty
while Q is not empty **do**
  $\{c_i \dots c_j\} := Q.poll()$ //choose an element of Q
  $neighbours := expand(\{c_i \dots c_j\})$ //generate neighbours according to the Variable-Based Neighbourhood
  for all $c_k \in neighbours$ **do**
    if $\{c_i \dots c_j\} \cup c_k$ is satisfiable **then**
      $Q.add(\{c_i \dots c_j\} \cup c_k)$
    else
      $MUS.add(\{c_i \dots c_j\} \cup c_k)$
    end if
  end for
end while

As an improvement upon the technique presented in (Gasca *et al.*, 2007) when it comes to calculate the neighbours of a set of business compliance constraints, not all the neighbours are generated: an order is established between the constraints, so that only those neighbours that are subsequent to all the constraints in the set are generated. In this way we succeed in generating only *new* neighbours, thereby avoiding the repeated study of the satisfiability of the same collection of constraints.

Once all *MUSes* have been obtained, each *MUS* is then associated to the activities related to the constraints that it contains. That is, for every *MUS* obtained, we count on a set of activities related to that *MUS*, in such a way that the activity or activities responsible for the unexpected behaviour is contained in those groups. As in the technique based on reified constraints, the minimal hitting sets (Definition 4) are used to find out that activities.

The calculation of these minimal hitting sets of activities provides us with the minimal diagnosis for a business process instance.

6 EXPERIMENTAL RESULTS

In this section, the results obtained in both proposals are presented. In order to do that, and due to the temporal complexity of the diagnosis using reified constraints is negligible (always lower than 1 millisecond), only the temporal complexity of both algorithms for obtaining the *MUSes* in Subsection 5.2 are compared. Moreover, the minimal diagnosis results obtained by the two techniques are presented.

Both techniques to calculate the *MUSes* obtain the same diagnosis results, and do not present any false positive. Therefore, Table 1 presents only the computational complexity comparison. The tests have been performed using different test cases with different number of constraints. The table shows the average time spent by each technique for each test case and the iterations performed (that is, the number of subsets whose satisfactibility is checked). It is possible to notice that the exhaustive technique spends rather more time than the Variable-Based Neighbourhood technique. The reason for this is that the exhaustive technique tries to find the *MUSes* by checking all the possible subsets of constraints, whereas the Variable-Based Neighbourhood technique is confined to the subsets formed by neighbours.

Table 1: Comparison between MUSes algorithms

#business compliance constraints	Exhaustive technique		Variable-Based Neighbourhood technique	
	time (ms)	iter.	time (ms)	iter.
9	2,588,891	502	12,547	15
9	1,031,110	256	15,437	24
9	1,004,750	256	12,953	15
9	336,000	129	63,844	16
9	114,265	67	48,235	16
9	114,703	67	46,531	12
11	3,354,115	951	32,313	37
11	426,234	131	10,109	16
11	742,188	157	17,438	22
13	2,326,453	514	15,156	21
13	3,004,571	725	21,063	31

Table 2 shows the different diagnoses obtained from the different techniques, using different test cases for the example of business process shown in Figure 1. Those tests are developed from a satisfiable solution,

changing some input data in order to cause inconsistencies:

- Test case 1: Establishing a very low quantity of accepted papers (*NumAccPaper* = 10).
- Test case 2: With only 60 participants (*NumPT* = 60).
- Test case 3: Not receiving much money from the partners, and spending too much money in inviting an international speaker (*partnerPart* = 2001, *invitedTalk* = 4250).

Table 2: Diagnosis results

test case	Minimal Diagnosis
1	Accepted Paper Request
2	Registration
3	International Invited Talk

As mentioned above, both techniques find out the minimal diagnosis. Regarding the diferencies in the temporal complexities, it is due to the search strategy followed by each technique. On the one hand, the technique based on reified constraints is better when it comes to find single faults. On the other hand, the technique based on the attaintment of the *MUSes* performs a search tree that first reaches the solutions with multiple faults. Therefore, the technique to use must be selected depending on the kinds of faults (single or multiple) that use to appear in the process to diagnose.

7 CONCLUSIONS AND FUTURE WORK

In this work, a new framework for the diagnosis of business processes is presented. This framework permits to diagnose a business process instance in function of a set of business compliance rules and for input and output data. The business compliance rules are described by constraints and are stored in a Constraint Database. Two types of techniques based on constraint programming are herein defined for the attainment of the minimal diagnosis in an efficient way, and considering the activities depending on the trace of the business process instance.

In this paper, only one checkpoint of the business process has been proposed, performing the diagnosis for only an instance. For future work, it may be possible to locate alternative check points, in accordance with the demand by companies, obtaining a run-time diagnosis of the business process, and inferring more information using several instances.

ACKNOWLEDGMENTS

This work has been partially funded by the Junta de Andalucía by means of 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

(de la Banda *et al.*, 2003) Maria J. García de la Banda, Peter J. Stuckey, and Jeremy Wazny. Finding all minimal unsatisfiable subsets. In *PPDP*, pages 32–43. ACM, 2003.

(Gasca *et al.*, 2007) R. M. Gasca, C. Valle, M. T. Gómez-López, and R. Ceballos. Nmus: Structural analysis for improving the derivation of all muses in overconstrained numeric csps. pages 160–169, 2007.

(Ghose and Koliadis, 2007) Aditya Ghose and George Koliadis. Auditing business process compliance. In *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing*, pages 169–180, Berlin, Heidelberg, 2007. Springer-Verlag.

(Gómez-López *et al.*, 2009) María Teresa Gómez-López, Rafael Ceballos, Rafael M. Gasca, and Carmelo Del Valle. Developing a labelled object-relational constraint database architecture for the projection operator. *Data Knowl. Eng.*, 68(1):146–172, 2009.

(Guillou *et al.*, 2009) Xavier Le Guillou, Marie-Odile Cordier, Sophie Robin, and Laurence Roze. Monitoring ws-cdl-based choreographies of web services. In *Proceedings of the 20th International Workshop on Principles of Diagnosis*, pages 43–50, June 2009.

(Kanellakis *et al.*, 1992) Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. Constraint query languages, 1992.

(Namiri and Stojanovic, 2007) Kioumars Namiri and Nenad Stojanovic. A semantic-based approach for compliance management of internal controls in business processes. In Johann Eder, Stein L. Tomassen, Andreas L. Opdahl, and Guttorm Sindre, editors, *CAiSE Forum*, volume 247 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

(Reiter, 1987) R Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

(Sadiq *et al.*, 2007) Shazia Wasim Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors, *BPM*, volume 4714 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2007.

(Weber *et al.*, 2009) Barbara Weber, Shazia Wasim Sadiq, and Manfred Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.

(Weske, 2007) M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, 2007.