Hybrid Diagnosis Applied to Multiple Instances in Business Processes

Rafael Ceballos(✉), Diana Borrego, María Teresa Gómez-López, and Rafael M. Gasca

Universidad de Sevilla, Seville, Spain
{ceball,dianabn,maytegomez,gasca}@us.es

Abstract. Business Process compliance is an important issue in control-flow and data-flow perspectives. Control-flow correctness can be analysed at design time, whereas data-flow accuracy should be verified at run-time, since data is accessed and modified during execution. Compliance validation should consider the conformance of data to business rules. Business compliance rules are policies or statements that govern corporate behaviour. Since business compliance rules and data change during process execution, faults can appear due to the erroneous inclusion of rules and/or data in the process. A hybrid diagnosis therefore needs to be performed regarding the likelihood of faults in data vs. business rules. In order to achieve the correct diagnosis, it is fundamental to attain the best assumption concerning the degree of likelihood. In this paper, we present an automatic process to diagnose possible faults that simultaneously combines business rules and data of multiple process instances. This process is based on Constraint Programming paradigm to efficiently ascertain a minimal diagnosis. Furthermore, a methodology for calculation of the most appropriate degree of likelihood of faults in data vs. business rules is proposed.

Keywords: Business process analysis · Diagnosis · Business rules · Business data constraints · Constraint programming · Databases

1 Introduction

Business processes (BPs) permit the description of the activities necessary to achieve an objective in a company. This description includes, among other things: a workflow model, a set of business rules or policies, and the data interchanged during the execution. The correctness of an execution implies the correctness of these three aspects. Frequently, BPs are supported by Process Aware Information Systems (PAISs) [23]. A PAIS is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models. This type of system provides a way to manage data stored in a repository layer that is read and written by a BP. The diagnosis of the workflow tends to be performed at design time to prevent errors after process deployment. However, the updating of business rules, such as

compliance rules, and the management of data at runtime is common practice. Since data and compliance rules may be modified at runtime, their diagnosis cannot be included in only the design phase.

The development an efficient diagnosis of possible faults is essential, since these faults appear at runtime and a great quantity of data and business compliance rules are probably involved. The special problem faced in BP diagnosis is that the data is not involved in only one instance, and it is not isolated from other instances executed in the past or in the future. The data written or read in an instance can be shared with other instances, or even with another process, such as when the data involved is stored in a repository, typically in a relational database. These relations must be used in the diagnosis process, since the isolation of a fault that explains a failure of an instance cannot contradict the diagnoses found for other instances.

In previous work [8], the importance of data correctness in BPs is studied, including relational databases as the main source of data. This previous work, however, only includes the possibility of faults in data, but fails to consider defects in business compliance rules. Derived from the modification of business compliance rules and data, certain faults can be produced due to the erroneous inclusion of the rules and/or data in the process. The importance of verifying the correctness of data in PAISs is known [12], although how to combine faults in business compliance rules and data at the same time remains a challenge. In this paper, we propose an automatic model-based diagnosis methodology to verify the compliance to the business rules by data in multiple instances, and to isolate the origin of the faults. Since data is more numerous and even more frequently updated, it is more likely a fault appearing due to incorrect data than due to an erroneous business compliance rule. In order to obtain the best assumption about this degree of likelihood, we propose a methodology for calculating the most appropriate degree of likelihood of faults between data vs. business compliance rules.

The paper is organised as follows. Section 2 presents a motivating example to illustrate the concepts. Section 3 introduces the adaptation of model-based diagnosis methodology. Section 4 presents the Constraints Programming paradigm used to perform the diagnosis. In Sect. 5, the diagnosis of the motivating example is performed. Section 6 presents an overview of related work found in the literature. And finally, conclusions are drawn and future work is proposed.

2 Using Business Data Constraints. A Motivating Example

In this paper, a real example of a financial economic application is used to illustrate the hybrid diagnosis. The activities of the company are oriented towards negotiating collaborative projects developed over a number of years. The process consists of the management of costs of projects, during their execution, as detailed in [8], and is represented in Fig. 1. All these tasks are carried out by a total of 25 employees, who modify the stored information for more than

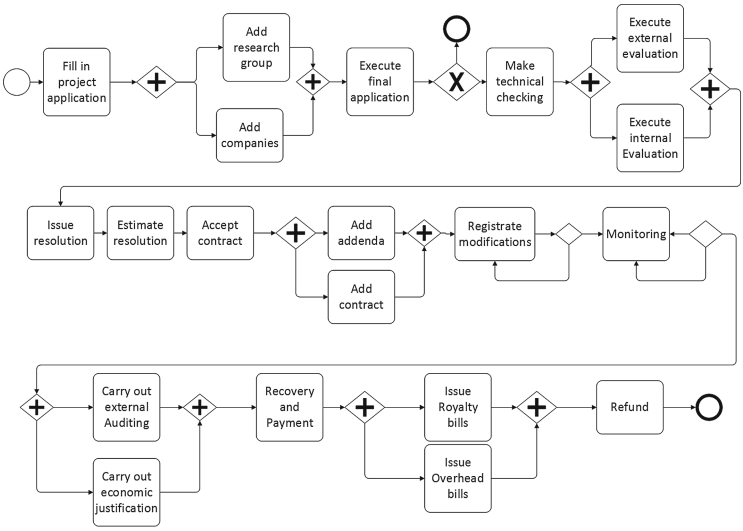

Fig. 1. Business Process example

300 projects. Each employee is responsible for certain activities of the process. The persistence layer that supports this business process is formed of a database with 86 tables. Each employee can introduce an average of 200 items of data per project, during the 4 years that a project can last.

BPs with a high level of human interaction demand a more frequent data validation and diagnosis, since humans are more likely to introduce intermittent faults. The intermittence of faults complicates the detection and diagnosis of model violations, since an inconsistency detected during the execution of an activity does not necessarily imply a failure in the activity, neither does it imply that this fault may appear again in the future. Our paper focuses on the concept of Business Data Constraints (BDCs), which was introduced in [8].

Definition 1. *Business Data Constraints are a subset of business compliance rules that represent the compliance relation between the values of data during a business process instance.*

In this paper, we assume that the BDCs specification can be incorrect, and therefore inconsistent with the introduced data, where the data is correct. The use of BDCs hugely facilitates the data consistency analysis, and the diagnosis of the origin of an inconsistency. The diagnosis methodology must consider the following characteristics:

- Data involved in the diagnosis is not strictly flowing through the process, some of them may also be stored in databases. This implies that the quantity of data involved in the diagnosis of each instance can be very large.
- The data managed in an instance is not independent from the data of other instances, since data can be shared between instances.

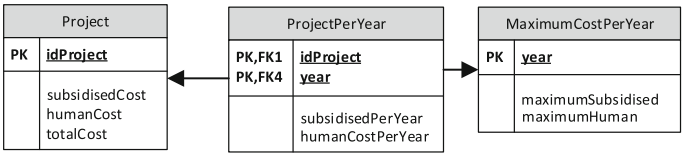

Fig. 2. Subset of relational model of the example

Project

idProject	subsidised Cost	human Cost	total Cost
223	55000	25000	80000
224	60000	26000	88000
...	...	...	...

ProjectPerYear

idProject	year	subsidised PerYear	humanCost PerYear
223	2015	7000	8000
223	2016	16000	9000
224	2015	8000	9000
224	2016	18000	6000
224	2017	9000	8000
...	...	...	...

MaximumCostPerYear

year	maximum Subsidised	maximum Human
2015	10000	10000
2016	10000	9000
2017	15000	12000
...	...	...

Fig. 3. Subset of tables of the example

– The BDCs tend to be updated in order to represent new conditions. Therefore, it is necessary to include these modified rules as possible faults.

The activities of a BP can modify certain data from a relational database, thereby making it necessary to evaluate certain BDCs. For the example, the BDCs must be satisfied with the various values of the project data. In order to express the BDCs, we use the grammar proposed in [8], which is based on numerical constraints over natural, integer, and float variables. A subset of BDCs for the activities is presented below:

1. Execute final application:
   (a) humanCost + subsidisedCost = totalCost (BDC_1)
   (b) $3 \cdot$ humanCost $\leq$ totalCost (BDC_6)
2. Accept contract:
   (a) subsidisedCost $\geq 2 \cdot$ subsidisedPerYear (BDC_2)
   (b) humanCost $\geq 4 \cdot$ humanCostPerYear (BDC_3)
3. Recovery and payment:
   (a) subsidisedPerYear $\leq$ maximumSubsidised (BDC_4)
   (b) humanCostPerYear $\leq$ maximumHuman (BDC_5)

The variables in the previous BDCs are stored in a database whose relational model is shown in Fig. 2. The three tables represent the information about the project (*Project*), the details for each project in each year *ProjectPerYear*, and the maximum spending limit allowed in each year and for each cost item (*MaximumCostPerYear*). Examples of the stored data are shown in Fig. 3.

In order to show our diagnosis methodology, in this example, we introduce 3 defects: 2 erroneous items of data, and 1 incorrect BDC. The following section shows how to adapt model-based diagnosis to the hybrid problem.

3 Applying Model-Based Diagnosis to Business Processes

3.1 Fundamentals of Model-Based Diagnosis

Model-based diagnosis enables the identification of the parts that fail in a system. It is performed by comparing the expected behaviour of the system with real behaviour. The expected behaviour is modelled using the knowledge of the system to diagnose, whereas real behaviour is known by analysing the events produced. This implies that model-based diagnosis is considered by the pair $\{SD, OM\}$, where SD is the System Description and OM is the Observational Model. The SD is a set of constraints, and the OM is a set of values of the observable data. A fault is visible when a discrepancy between the expected behaviour (SD) and the observed behaviour (OM) is found.

Model-based diagnosis is based on the parsimony principle [17], in order to attain a minimal diagnosis that explains the conflicts in an efficient way. This principle states that among competing hypotheses, the one with the fewest assumptions should be selected. For example, a conflict is detected in the following SD and OM: $\{a + b = c,\ a + 2{\cdot}b = d\}$, $\{a = 7,\ b = 5,\ c = 9,\ d = 14\}$. If the assumption $\{a = 7\}$ is false (i.e. it is modified, $a = 4$), then the remaining assumptions are satisfied between them. Model-based diagnosis identifies the smallest assumption that causes conflicts. In this example, there are other possible hypotheses, but these imply the modification of more than one assumption.

The following subsections analyse: (1) how to design the model to be diagnosed (SD); and (2) how to obtain the observational model (OM).

3.2 System Description: Relational Database Model and Business Data Constraints

Since the stored data participates in the BDCs, it is necessary to include the BDCs and the relational database scheme into the model to be diagnosed. Business Data Constraints describe the semantic relation between the data values that are introduced, read and modified during the BP instances. It should be borne in mind that the variables participating in the constraints can come from the database or from the data-flow.

A Relational Database is a collection of predicates over a finite set of variables described by means of a set of relations. A relation R is a data structure which consists of a heading and an unordered set of tuples which share the same type, where A_1, A_2, ..., A_n are attributes of the domains D_1, D_2, ..., D_n. A number of the attributes of a relation can be described as *Primary Key Attributes*. The relation between two tables is described by a referential integrity. Two tables can be related by means of their *Primary* and *Foreign Key Attributes*, described in the literature as the relational model.

3.3 Observational Model: Tuples of the Database

In a PAIS, the information is typically stored in a relational database, and therefore the tuples of the tables compose the OM. Since the variables involved in

BDCs:		(1,2)		(1,3,6)		(1,6)			(2,4)		(3,5)		(4)		(5)	
tuple	idProj.	Subs. Cost Var.	Subs. Cost	Hum. Cost Var.	Hum. Cost	Tot. Cost Var.	Tot. Cost	Year	Subs. PerYear Var.	Subs. Per Year	Hum. PerYear Var.	Hum. Per Year	Max. Subs. Var.	Max. Subs.	Max. Hum. Var.	Max. Hum.
1	223	Subs. Cost1	55000	Hum. Cost1	25000	Tot. Co1	80000	2015	Subs. PerYear1	7000	Hum. PerYear1	8000	Max. Subs1	10000	Max. Hu1	10000
2	223	Subs. Cost1	55000	Hum. Cost1	25000	Tot. Co1	80000	2016	Subs. PerYear2	16000	Hum. PerYear2	9000	Max. Subs2	10000	Max. Hu2	9000
3	224	Subs. Cost2	60000	Hum. Cost2	26000	Tot. Co2	88000	2015	Subs. PerYear3	8000	Hum. PerYear3	9000	Max. Subs1	10000	Max. Hu1	10000
4	224	Subs. Cost2	60000	Hum. Cost2	26000	Tot. Co2	88000	2016	Subs. PerYear4	18000	Hum. PerYear4	6000	Max. Subs2	10000	Max. Hu2	9000
5	224	Subs. Cost2	60000	Hum. Cost2	26000	Tot. Co2	88000	2017	Subs. PerYear5	9000	Hum. PerYear5	8000	Max. Subs3	15000	Max. Hu3	12000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Fig. 4. Denormalized tuples

a BDC have different origins, it is possible that attributes from various tables are related in a single BDC. The location of the data in various tables is due to the necessity to follow the *Normal Forms* defined in relational database theory. The normalization rules are designed to prevent update anomalies and data inconsistencies. Since data is stored in various tables, to ascertain the full tuple of values for an OM, a *denormalization process* needs to be carried out. This denormalization process is only used for the purpose of diagnosing; the relational database undergoes no changes at all, only a new join relation is obtained with all related attributes together used for the diagnosis in a temporal way. Although normalization methods are applied at schema level, it is possible to apply the opposite methods to ascertain the denormalized relations between the data. The related attributes are those that appear in the same BDC together with the primary-foreign key attributes necessary to join the related attributes that belong to various tables. For the example in Fig. 2, the obtained join-table is shown in Fig. 4. In the figure, the related BDCs are shown at the top of each column. Although the details of how the join-table is created are described in [8], the general idea can be understood by analysing the relational model in Fig. 2, and by observing the specific values for the example introduced in Fig. 3. Since the attribute *idProject* of the table *ProjectPerYear* is a foreign key of table *Project*, then each tuple of *ProjectPerYear* is related to the tuple of *Project* whose foreign and primary keys are equal. In a similar way, these are relation between the variables *MaximumPerYear* and *ProjectPerYear*.

In the denormalization process, a column is included for each attribute to distinguish between the provenance of the values. Distinction is made due to the fact that it is necessary to know whether two values correspond to the same attribute after the denormalization, since two equal values in a column do not imply they represent the same variable. In the denormalization process, the same value can appear in various tuples, derived from the 1..n relation between the tables, such as the value associated to the human cost of project 223 (Variable *humanCost1*) that appears in the two first tuples of Fig. 4, since this project was developed over two years.

4 Constraint Programming for Hybrid Business Process Diagnosis Models

4.1 Fundamentals of Constraint Programming

In order to find the minimal incorrect part of the SD and OM in an automatic and efficient way, we propose using Constraint Programming, since the definition of BDCs is very close to the definition of the logic and arithmetic constraint modelled in a Constraint Satisfaction Problem (CSP). A CSP [18] represents a reasoning framework consisting of variables, domains and constraints $\langle V, D, C \rangle$, where V is a set of n variables $v_1, v_2, ..., v_n$ whose values are taken from finite domains $D_{v1}, D_{v2}, ..., D_{vn}$ respectively, and C is a set of constraints on their values. The constraint c_k $(x_{k1}, \ldots, x_{kn})$ is a predicate that is defined on the Cartesian product $D_{k1} \times \ldots \times D_{kj}$. This predicate is true iff the value assignment of these variables satisfies the constraint c_k.

If the model represented by $\{SD, OM\}$ is satisfiable, then the OM conforms to the BDCs that describe the SD. However, if no solution is found, the minimal non-conformance parts of the model should be determined. In order to ascertain the minimal explanation regarding faults, it is necessary to find a minimal subset $ss \subset \{\text{OM} \cup \text{BDCs}\}$ that satisfies $\{\text{SD} \cup \text{OM}\} - ss$. Since BDCs are applied by various tuples with different values, all instances of BDC_i must be included in the search. This search is performed using a Constraint Optimization Problem (COP), which is solved as a Min-CSP. A Min-CSP is a COP, where the goal is to minimize an optimization function. The application of a Min-COP to a model-based diagnosis problem implies defining an optimization function in order to minimize the subset ss.

4.2 Compliance Verification by Means of the Observational Model

In order to verify the compliance of the BDCs, we have applied the obtained tuples to instantiate the BDCs. For the tuples shown in Fig. 4, the BDCs of the example have been instantiated, and the results are shown in Table 1. Analysing each BDC according to the tuples:

Table 1. Results of the compliance verification of the BDCs

BDC	Tuple				
	1	2	3	4	5
1	BDC_1^1 ✓			BDC_1^2 ✗	
2	BDC_2^1 ✓	BDC_2^2 ✓	BDC_2^3 ✓	BDC_2^4 ✓	BDC_2^5 ✓
3	BDC_3^1 ✗	BDC_3^2 ✗	BDC_3^3 ✗	BDC_3^4 ✓	BDC_3^5 ✗
4	BDC_4^1 ✓	BDC_4^2 ✗	BDC_4^3 ✓	BDC_4^4 ✗	BDC_4^5 ✓
5	BDC_5^1 ✓	BDC_5^2 ✓	BDC_5^3 ✓	BDC_5^4 ✓	BDC_5^5 ✓
6	BDC_6^1 ✓			BDC_6^2 ✓	

- BDC_1: two different instances of this BDC can be obtained (BDC_1^1 and BDC_1^2) for the five tuples ({1, 2, 3} and {4, 5}). BDC_1^1 is satisfiable (tuples 1, 2 and 3), but BDC_1^2 is non-compliant (tuples 4 and 5). An error in one single input: *humanCost2*, *subsidisedCost2* or *totalCost2*, can explain this abnormal behaviour. It is less likely that there is a fault in a BDC (usually written by a business expert), than in an input (usually typed in by an user).
- BDC_2, BDC_5 and BDC_6 are consistent for all tuples.
- BDC_3: Only BDC_3^4 is satisfiable. There are several explanations for this behaviour: (a) related to the data values, and (b) related to the BDC. An error committed on writing BDC_3 could provide a single explanation. As mentioned earlier, it is less likely that a BDC is erroneous than data. The question is, in what percentage should this likelihood be described? The analysis of this percentage of likelihood is part of our methodology and is detailed in the following subsection.
- BDC_4: BDC_4^2 and BDC_4^4 are not satisfied. The failure can be explained with a single variable,{*maximumSubsidised2*}, whose value is 10000 and is shared by these two instances.

Although the possible diagnosis has been explained separately for each BDC_i, it is necessary to ascertain the minimal diagnosis that explains all discrepancies for the whole problem. Our methodology is able to obtain the minimal diagnosis that explains all this non-compliant behaviour.

Variables of the problem:
integer subsidisedCost1, subsidisedCost2, humanCost1, humanCost2, ...
integer[0,1] rSc1, rSc2, rHc1, rHc2, rTc1, rTc2, rHp1, rHp2, rHp3, rHp4, rHp5, ..., $rBDC_1$, $rBDC_1^1$, $rBDC_1^2$, $rBDC_2$, $rBDC_2^1$, ..., $rBDC_5^5$, $rBDC_6$, $rBDC_6^1$, $rBDC_6^2$
Constraints to represent the instantiation of Variables:
$rSc1 = \neg(subsidisedCost1 = 55000)$ | $rSc2 = \neg(subsidisedCost2 = 60000)$
$rHc1 = \neg(humanCost1 = 25000)$ | ...
Constraints to represent the instantiation of BDCs:
$rBDC_1^1 = \neg(humanCost1 + subsidisedCost1 = totalCost1)$
$rBDC_1^2 = \neg(humanCost2 + subsidisedCost2 = totalCost2)$
$rBDC_2^1 = \neg(subsidisedCost1 \geq 2 \cdot subsidisedPerYear1)$
...
$rBDC_1 = (rBDC_1^1 + rBDC_1^2 \geq minLik_1)$
$rBDC_2 = (rBDC_2^1 + rBDC_2^2 + rBDC_2^3 + rBDC_2^4 + rBDC_2^5 \geq minLik_2)$
...
$((rBDC_1^1 + rBDC_1^2 = 0) \vee (rBDC_1^1 + rBDC_1^2 \geq minLik_1)$
$(rBDC_2^1 + rBDC_2^2 + rBDC_2^3 + rBDC_2^4 + rBDC_2^5 = 0) \vee$
$(rBDC_2^1 + rBDC_2^2 + rBDC_2^3 + rBDC_2^4 + rBDC_2^5 \geq minLik_2)$
...
Objective function:
$minimize(rSc1 + rSc2 + rHc1 + rHc2 + rTc1 + rTc2 + ... + rBDC_1 \cdot minLik_1 + rBDC_2 \cdot minLik_2 + ... + rBDC_6 \cdot minLik_6)$

Fig. 5. Min-CSP example

4.3 Min-CSP Applied to Model-Based Diagnosis

The execution of the diagnosis entails the translation of the problem into a CSP, including BDCs and tuples for each execution instance. Figure 5 shows the Min-CSP created to diagnose the example. Below, the modelling of the parts of the CSP are detailed (variables, domains, constraints and objective function).

In order to declare the **Variables of the problem**, a new variable is added to the Min-CSP for each variable obtained in BDC_j^i as explained in Subsect. 3.3 following the syntax: *type* var_k^1,...,var_k^m.

Furthermore, in order to provide the Min-CSP with the ability to distinguish between different sources of faults (i.e. data and/or BDCs), we use reified constraints. A reified constraint relies on a variable that denotes its truth value. It is therefore necessary to add new variables to the CSP, whose domain is reduced to values 0 (false value) and 1 (true value). These variables are associated to each BDC_i, BDC_i^j (**Constraints to represent the instantiation of BDCs**) and assignments of value to an input (**Constraints to represent the instantiation of Variables**), in order to denote whether they are satisfiable. Both $rBDC^i$ and $rBDC_i^j$ are included to differentiate the BDC from its application for each tuple. The proposed syntax is:

//Reified variables to ascertain the satisfiability of the BDCs
integer[0,1] rVar_k^1,...,rVar_k^m
integer[0,1] rBDC_i, rBDC_i^1,...,rBDC_i^n
//Constraints to represent the reified variables assignment
$\text{rVar}_k^j = \neg(\text{var}_k^j = \text{value}_k^j)$
//Constraints to represent the reified BDCs
$\text{rBDC}_i^1 = \neg(\text{BusinessRule}_i$ instantiated by tuple 1)
...
$\text{rBDC}_i^n = \neg(\text{BusinessRule}_i$ instantiated by tuple n)

The reified variables are equalized to the negated constraints since the objective function is to minimize the number of elements with abnormal behaviour (non-compliant). In order to ascertain when a defect in a BDC is less likely than in data errors, the following constraint is added for each BDC:

$$\text{rBDC}_i = \text{rBDC}_i^1 + ... + \text{rBDC}_i^n = \sum_j^n \text{rBDC}_i^j \geq \text{minLik}_i$$

These constraints incorporate the likelihood concept into the CSP, by using the parameter minLik_i.

Definition 2. *The parameter* $minLik_i$ *is the minimum number of faults (non-compliant instances of a* BDC_i*) that is set as the threshold to indicate that there is a defect in a* BDC_i*.*

For example, if there are 5 tuples where BDC_i is involved, $minLik_i$ can take a value between 1 and 5. If at least the $minLik_i$ threshold number of instances are

not satisfiable, then BDC_i is considered as a part of the minimal diagnosis. How the values of each $minLik_i$ is determined is detailed in the following subsection.

The **Objective function** is defined as:

$$\text{minimize}(\text{rVar}_k^1 + ... + \text{rVar}_k^m + ... + \text{rBDC}_1 \cdot \text{minLik}_1 + ... + \text{rBDC}_q \cdot \text{minLik}_q)$$

Each rBDC_i has a weighting that is proportional to each parameter $minLik_i$. This objetive implies finding the minimal hybrid diagnosis.

Finally, it is important to add this constraint for each BDC_i:

$$(\text{rBDC}_i^1 + ... + \text{rBDC}_i^n = 0) \vee (\text{rBDC}_i^1 + ... + \text{rBDC}_i^n \geq \text{minLik}_i)$$

This constraint permits two options: (1) $\sum_j^n \text{rBDC}_i^j$ is equal to 0, and therefore the BDC_i is correct; or (2) $\sum_j^n \text{rBDC}_i^j$ is equal to or greater than minLik_i, and therefore the BDC_i has a defect. Intermediate values between 0 and minLik_i are not allowed, and therefore, if there are inconsistencies in a BDC_i^j, it can only be avoided by relaxing the variables rVar_q related to BDC_i^j. In other words, the defects are only in input data and the BDC_i is correct.

4.4 Calculation of the MinLik Parameter

The appropriate value of the minLik parameter for each BDC_i depends on several factors. It is necessary to take into account the number of tuples and variables affected by each BDC, therefore *minLik* should be calculated at runtime, when the diagnosis process is performed. The calculations for the example are shown in Table 2. The meaning of each column is as follows:

Table 2. *minLik* parameter calculation

BDC	nVar	nInst	%errors	nErrors	domain	media	rep?	cover	reduced	minLik
1	3	2	20 %	$1.2 \approx 1$	[1]	1	No			2
2	2	5	20 %	2	[1, 2]	2	Yes	2	1	1
3	2	5	20 %	2	[1, 2]	2	Yes	2	1	1
4	2	5	20 %	2	[1, 2]	2	Yes	3	2	2
5	2	5	20 %	2	[1, 2]	2	Yes	3	2	2
6	2	2	20 %	$0.8 \approx 1$	[1]	1	No			2

- *BDC*: number that identifies a BDC_i.
- *nVar*: number of variables involved in a BDC_i.
- *nInst*: number of instances ($\text{BDC}_i{}^j$) of the BDC_i.
- *%errors*: average percentage of data errors estimated by the expert.
- *nErrors*: probable number of data errors that can appear in all instances of a BDC_i. This is obtained as the product of: $nErrors = nVar \cdot nInst \cdot \%errors$, and it is rounded to the nearest integer.

– *domain*: interval between the minimum and maximum number of instances that can be influenced by a number of data errors equal to $nErrors$.
– *media*: most likely case within the range obtained in the previous *domain* column. It is calculated as the weighted average of all possible cases (integers of the *domain* column) and probability, rounded to the nearest integer.
– *rep*?: column set to Yes if there is at least one variable that appears in different instances of the same BDC_i. For example, humanCost2 appears in BDC_3^3, BDC_3^4, BDC_3^5.
– *cover*: minimum number of items of erroneous data such that all ${BDC_i}^j$ of a BDC_i are incorrect. This depends on the existence of variables that participate in different instances of the same BDC_i (previous column *rep*?). For example, for BDC_3, the minimal number is two: variables humanCost1 and humanCost2.
– *reduced*: the diagnosis process tries to explain the anomalous behaviour with the minimum number of errors, but the variables that are repeated in different instances can explain errors in several instances at the same time. To counteract this effect, this column is calculated as $1+(media \cdot cover)/nInst$, and is rounded to the nearest integer.
– *minLik*: calculation performed as the minimum between the values *media*+1 and *reduced*.

4.5 Diagnosing the Example

The minimal diagnosis is obtained solving the Min-CSP presented in Fig. 5. The minimal diagnosis is a set of three elements: maximumSubsidised2, BDC_3, and {humanCost2, subsidisedCost2, or totalCost2}. In order to satisfy all BDCs and the compliance verification presented in Table 1, three modifications could be made:

– The input data associated to the $maximumSubsidised2$ variable must be changed. This change solves the compliance problems in BDC_4.
– The BDC_3 must be changed. This change solves the compliance problems in BDC_3.
– Finally, the input data associated to, $humanCost2$ or $subsidisedCost2$ or $totalCost2$, must be changed. Only one of three variables. This change solves the compliance problems in BDC_1.

The minimal diagnosis found with our methodology includes the three introduced faults.

5 Evaluation

In order to evaluate our proposal, we have designed a set of tests where the possible single and double faults in data and BDCs are simulated. With these tests, we can confirm the validity of our methodology to achieve the minimal

diagnosis for various cases. To cover the most relevant cases, variables and BDCs of the example have been divided into different sets according to the number of tuples and BDCs that are affected by each of them. Regarding the 22 variables in Fig. 4, we have divided them into 7 different sets, where each set is formed of the variables that appear in the same number of tuples and involved in the same BDCs. Regarding BDCs, the 6 BDCs of the example are divided into 4 sets according to the number of variables affected by them. In greater detail, the two sets of variables and BDCs are:

- **Set of Variables:** {subsidisedCost1, 2, 3, 4, 5, maximumHuman3, maximumSubsidised3} are in one tuple and affected by one BDC; {humanPerYear1, 2, 3, 4, 5} are in one tuple and affected by three BDCs; {maximumSubsidised1, 2, maximumHuman1, 2} are in two tuples and affected by one BDC; {subsidisedCost1, totalCost1} in two tuples and affected by two BDCs; {HumanCost1} is in two tuples and affected by three BDCs; {subsidisedCost2, totalCost2} are in three tuples and affected by two BDCs; {humanCost2} is in three tuples and affected by three BDCs.
- **Set of BDCs:** {BDC_6} involves four variables, {BDC_1} involves six variables, {BDC_2, BDC_3} involve seven variables, {BDC_4, BDC_5} involve eight variables.

Figure 6 depicts the execution time (ms) for the proposed hybrid diagnosis. Each possible fault is simulated for an example of each set of variables and BDCs, since the study for the elements within the same set is equivalent. In the figure, different symbols are used to represent the results of the attained diagnosis: (1) green diamonds, the introduced fault is the minimal and single fault found with our methodology; (2) blue squares, the introduced fault is one of the minimal faults found by our methodology; and (3) red triangles, the introduced fault is not minimal, and hence our approach found another minimal explanation.

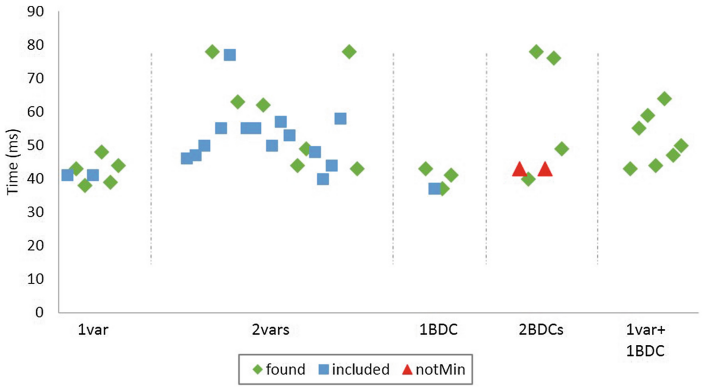

Fig. 6. Execution time of hybrid diagnosis test cases

In these forty-five tests, only in two cases did our approach find another minimal explanation. In these two cases, the correct diagnosis is not the minimal diagnosis, but is proposed as the second minimal diagnosis.

6 Related Works

Papers where data is involved in model-based diagnosis in BPs are divided into two types: model analysis at design time, and analysis of instances at runtime.

Regarding the analysis of the data model at design time, one of the main focuses is on the detection of possible faults in the data flow, such as missing, redundant, and conflicting data [21]. This research has been extended to deal with the analysis of process models that contain both control flow and data flow, and with artifact-centric orientation instead of activity-centric models [7,10,20]. Although a variety of mechanisms have been developed to prevent errors at the structural level (deadlocks, livelocks, ...) [22], they also have to comply with business level rules and policies. In [5,6], the activities are attributed with pre-conditions and post-conditions that describe the data behaviour to verify the correctness of the model at design time. Artifact-centric orientation has also been used to support consistent specifications [26].

Regarding the importance of data for the runtime conformity of BPs, both stored data and data flow are objects of this study. In relation to the persistence layer and data-flow, relational databases have been used in BPs, for example in [2], which presents a solution where data is audited and stored in a relational database. However, no validation of the semantics is performed for this persistence layer and the business rules.

The analysis of the correctness of BPs is typically related to the activity executed according to the value of a data variable in each case, by verifying whether the model and the log conform to each other [1]. Although some authors have noticed that relational databases are the typical repository where the changed data is stored instead of log events [3], the stored data itself does not represent the objective of the diagnosis. In [15,16], business constraint monitoring is presented based on Event Calculus. In [12], the importance of validating the correctness of data in PAISs is highlighted, although the challenge remains of how to find a fault in data instead of in a decision related to data. Therefore, conformance-checking analysis on log events is insufficient [19] to claim correctness in a BP.

In this paper, we present the necessity to study the correctness of rules and data themselves, and define data-aware compliance rules (BDCs). Related to how to model data-aware compliance rules, studies such as [4,11,12,14,24] define graphical notations to represent the relationship between data and compliance rules by means of data conditions. In [25], a method for monitoring control-flow deviations during process execution is proposed. In [13], "semantic constraints" and the SeaFlows framework for enabling integrated compliance support are proposed. An approach for semantically annotating activities with preconditions and effects that may refer to data objects is introduced in [9], and an efficient algorithm for compliance verification using propagation is also discussed.

Summarizing, to the best of our knowledge, this is the first contribution that addresses a hybrid approximation, where faults in rules and data are considered at the same time. Previous studies can be found in the literature about the *Possible Minimal Set of Incorrect Data* or *Possible Minimal Set of Incorrect BDCs*, but this work is centred on both types of errors at the same time. A preliminary study [8] diagnoses data stored according to the model, but not combined with possible faults in business rules.

7 Conclusions and Future Work

A diagnosis methodology that considers both business data constraints and data (either flowing or stored) as possibly being responsible for incorrect behaviour is presented. The combination of types of faults (i.e. in BDCs and/or data) necessitates a hybrid diagnosis, which is performed regarding the likelihood of faults in data vs. those in BDCs. To this end, Constraint Programming is used by modelling the problem as a CSP. Moreover, this proposal takes into account that data may be shared by various instances of the BP, and deals with it accordingly, for example by diagnosing faults in an instance that were caused by the updating of data by another running instance.

As future work, we plan to offer an easier and customized way to determine the likelihood between data and BDC malfunction. In order to improve our approach, we would like to consider roles or the organization view. Moreover, we intend to perform the diagnosis even when certain data still remains unknown, in order to allow the detection of potential errors in advance.

Furthermore, we would like to extend the idea to include those BPs that manage aggregate data. Another interesting line would be to manage a log of diagnoses, whereby the cause of a malfunction is ascertained by analysing previous diagnoses.

Acknowledgement. This work has been partially funded by the Ministry of Science and Technology of Spain (TIN2015-63502) and the European Regional Development Fund (ERDF/FEDER). Thanks to Lesley Burridge for revision of the English version of the manuscript.

References

1. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Int. Rev. Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
2. van der Aalst, W., van Hee, K., van der Werf, J.M., Kumar, A., Verdonk, M.: Conceptual model for online auditing. Decis. Support Syst. **50**(3), 636–647 (2011)
3. van der Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. Management for Professionals, pp. 105–128. Springer, Switzerland (2015)

4. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. J. Vis. Lang. Comput. **22**(1), 30–55 (2011)
5. Borrego, D., Eshuis, R., Gómez-López, M.T., Gasca, R.M.: Diagnosing correctness of semantic workflow models. Data Knowl. Eng. **87**, 167–184 (2013)
6. Borrego, D., Gasca, R.M., Gómez-López, M.T.: Automating correctness verification of artifact-centric business process models. Inf. Softw. Technol. **62**, 187–197 (2015)
7. Eshuis, R., Kumar, A.: An integer programming based approach for verification and diagnosis of workflows. Data Knowl. Eng. **69**(8), 816–835 (2010)
8. Gómez-López, M.T., Gasca, R.M., Pérez-Álvarez, J.: Compliance validation and diagnosis of business data constraints in business processes at runtime. Inf. Syst. **48**, 26–43 (2015)
9. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 5–17. Springer, Heidelberg (2009)
10. Hull, R.: Artifact-centric business process models: brief survey of research results and challenges. In: Tari, Z., Meersman, R. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1152–1163. Springer, Heidelberg (2008)
11. Liu, Y., Müler, S., Xu, K.: A static compliance-checking framework for business process models. IBM Syst. J. **46**(2), 335–362 (2007)
12. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 9–23. Springer, Heidelberg (2010)
13. Ly, L.T., Rinderle-Ma, S., Göser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. Inf. Syst. Front., 1–25 (2009)
14. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 82–99. Springer, Heidelberg (2011)
15. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 146–162. Springer, Heidelberg (2012)
16. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the event calculus. ACM TIST **5**(1), 17 (2013)
17. Peng, Y., Reggia, J.: Abductive Inference Models for Diagnostic Problem-Solving. Symbolic Computation. Springer, New York (1990)
18. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
19. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
20. Sidorova, N., Stahl, C., Trcka, N.: Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible. Inf. Syst. **36**(7), 1026–1043 (2011)
21. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Sheng, O.R.L.: Formulating the data-flow perspective for business process management. Inf. Syst. Res. **17**(4), 374–391 (2006)
22. Trčka, N., van der Aalst, W.M.P., Sidorova, N.: Data-flow anti-patterns: discovering data-flow errors in workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 425–439. Springer, Heidelberg (2009)

23. Weber, B., Sadiq, S.W., Reichert, M.: Beyond rigidity - dynamic process lifecycle support. Comput. Sci. - R&D **23**(2), 47–65 (2009)
24. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: SBPM 2008: 3rd International Workshop on Semantic Business Process Management at ESWC 2008, June 2008
25. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-based monitoring of process execution violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 182–198. Springer, Heidelberg (2011)
26. Yongchareon, S., Liu, C., Zhao, X.: A framework for behavior-consistent specialization of artifact-centric business processes. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 285–301. Springer, Heidelberg (2012)