Compliance validation and diagnosis of business data constraints in business processes at runtime

María Teresa Gómez-López [a,*], Rafael M. Gasca [a], José Miguel Pérez-Álvarez [b]

[a] *Department of Languages and Computer Systems, University of Seville, Spain*
[b] *Intelliment Security, Spain*

Keywords:
Business processes validation and diagnosis
Business data constraints
Persistence data
Constraint programming

A B S T R A C T

Business processes involve data that can be modified and updated by various activities at any time. The data involved in a business process can be associated with flow elements or data stored. These data must satisfy the business compliance rules associated with the process, where business compliance rules are policies or statements that govern the behaviour of a company. To improve and automate the validation and diagnosis of compliance rules based on the description of data semantics (called Business Data Constraints), we propose a framework where dataflow variables and stored data are analyzed. The validation and diagnosis process is automated using Constraint Program-ming, to permit the detection and identification of possibly unsatisfiable Business Data Constraints, even if the data involved in these constraints are not all instantiated. This implies that the potential errors can be determined in advance. Furthermore, a language to describe Business Data Constraints is proposed, for the improvement of user-oriented aspects of the business process description. This language allows a business expert to write Business Data Constraints that will be automatically validated in run-time, without the support of an information technology expert.

1. Introduction

A business process consists of a set of activities that are performed in coordination within an organizational and technical environment [53]. The basic idea of Business Process Models (BPMs) is the explicit representation of business processes with their activities and of the execution constraints between activities. The description of the model is often insufficient to fully describe the behaviour of the process, and hence business compliance rules are added to improve the capacity of process description. Many studies propose different taxonomies to classify business compliance rules [8,21,40,44]. Business compliance rules can be understood as conforming to a rule such as a specification, a policy or a standardized procedure that represent a natural step towards the inclusion of semantic requirements between business functionality and data. This paper focuses on the use of Business Compliance Rules [4] to describe the data semantics of a business process for the representation of the relations between data values: these we have named Business Data Constraints (henceforth referred to as BDCs). These Business Data Constraints are understood as a subset of business compliance rules which represent the semantic relation between the data of a business process instance. In this paper we assume that the BDCs specification is correct, but they can be inconsistent because the introduced data are incorrect.

* Corresponding author.
E-mail addresses: maytegomez@us.es (M.T. Gómez-López), gasca@us.es (R.M. Gasca), jmperez@intellimentsec.com (J.M. Pérez-Álvarez).
URLS: http://www.lsi.us.es/~mayte (M.T. Gómez-López), http://www.lsi.us.es/~gasca (R.M. Gasca), http://www.intellimentsec.com/ (J.M. Pérez-Álvarez).

An example of the use of BDCs is to represent that the *sale price* of a product is the *price* of the product multiplied by a percentage of *tax* that has to be paid to the government. The *price* of the product is greater than or equal to the *production cost* plus the *percentage gain*. The *price* of the product can change depending on special discounts, but can never be less than the *production cost*. The values of *production cost*, *price*, *tax*, etc. are introduced by hand into the system by different employees in the company, from different departments, at different moments, and for several products. All these manners to introduce data can cause the introduction of incorrect values that do not follow the policies of the company.

To validate the correctness of the data, it is necessary to include the Business Data Constraints in the description of a business process model, since none of the activities of a process can work correctly using incorrect data. Incorrect data can be produced by other activities or introduced by users in human tasks. It is also necessary to take into account that the information handled in a business process tends to be extensive, since organizations currently need to manage a great deal of data; data that are normally stored on a relational database. Therefore, the data involved in a business process and in the BDCs can form part of the dataflow (depending on the instance), and can be stored on a database (persistence layer).

Although certain papers have analyzed dataflow validation, to the best of our knowledge there are no proposals for the validation of the semantics of dataflow and stored data in terms of their values. To validate the BDCs in accordance with a relational database, we propose a model-based validation and diagnosis methodology. This methodology is used to ascertain whether the behaviour of a system is correct, and to identify the violated BDCs for the data involved. This identification is carried out by comparing the expected behaviour represented by BDCs (the model), with the observed values of the variables (the observational model). Based on these principles, the contributions of this paper are:

- *Enlargement of the business process model for the validation and diagnosis for the compliance of BDCs*. This implies describing the business process model, defining the BDCs and determining the activities where the validation is necessary, and including the relational database layer. To enable the business expert to perform these actions, we have developed a framework where all these steps can be fulfilled. In a preliminary study [17], the necessity to validate and diagnose the stored data by means of the compliance of rules was detected. In the current paper, we have included various and more complex relations between the tables of the database, and an analysis of the Referential Integrity as a graph, to perform the automatic creation of the joined tables and the data involved.
- *Description of BDCs*. To describe the BDCs, we have developed a language as a way to represent the correct values of the involved data objects in a process. This language allows the business expert to represent correct data semantics for dataflow and stored data. The BDCs can be defined as either an invariant of the process, and therefore associated to the whole process, or as a contract (pre or post-condition) for an activity or a set of activities. We propose an algorithm to extract only the data involved in the evaluation at runtime, to determine the clustering of BDCs, thereby minimizing the evaluation time.
- *Early identification of non-compliance of BDCs in runtime*. Since not all the BDCs have to be analyzed in the whole business process [36], for each instance and activity it is necessary to obtain the dataflow and tuples of the stored data involved in the process instance. These form the observational model for the described model in design time. Once the tuples involved in the instances are known, we propose using the Constraint Programming paradigm, to automate the validation and diagnosis process, since the use of Constraint Programming brings the advantage of an early identification of non-compliance in BDCs (even before all the values are instantiated). To the best of our knowledge, no commercial tool to date has included the capacity to evaluate the correctness of the data before all the data is known. However, our implementation incorporates the algorithms to this effect in a commercial BPMS.

This paper is organized as follows: Section 2 presents the most interesting aspects related to BDCs and the orientation towards Constraint Programming. Section 3 gives an example of a real business process where BDCs are necessary. Section 4 sets out the proposed business model by means of a framework that includes the database to store the BDCs, and the relational database with the information of the business process instances. Section 5 presents the validation and diagnosis algorithm to be executed to find out the unsatisfiable BDCs. The application to include the BDCs and the connector that relates the process and the validation algorithm are presented in Section 6. Section 7 presents an analysis of related work that exists in the area. Finally, conclusions and future work are presented.

2. Representation and treatment of business data constraints

Although no standard meaning of 'Business Compliance Rules' exists, a business compliance rule is generally understood as which conforms to a rule such as a specification, a policy or a standardized procedure. The majority of studies in this area address business compliance rules as the representation of a relative order between the various activities executed in a process [23].

Several language constructs can be used to design business compliance rules. Current business rule engines are based on the use of IF–THEN rules or their derived extensions of ECA rules (event-condition-action) or ECAA rules (event-condition-action-alternative). Some examples of the current rule engines are Drools, Fair Isaac Blaze Advisor, ILOG JRules and Jess. Another possibility is the Business Process Compliance Language (BPCL) [54], which defines inclusion, precedence, and existence conditions for business rules by means of Object Constraint Language (OCL) expressions, which specify correctness and compliance checks. That version of BPCL was improved in [4] to develop a semantic

approach to business rule management that allows for intuitive modelling and analysis of business process compliance. Unfortunately, these proposals are not based on the treatment of data values (dataflow and stored data), hence the correctness of data at run time cannot be validated with unknown values and combined with diagnosis processes. For this reason, a new business process compliance methodology to describe, validate and diagnose the correctness of the data semantics is necessary. According to [22,15], successful business process compliance implementation requires an integrated approach, reflecting the entire BPM lifecycle; should support compliance verification beyond simple control flow aspects; needs an intuitive graphical notation for compliance requirements that is also comprehensible for non-experts, and should support the application of semantic technologies for the definition, implementation and execution of automated compliance verification.

In order to cover all these characteristics, and to include the relation between data values in the process, we propose that BDCs be presented as Numerical Constraints, as detailed in the following section.

2.1. Grammar of business compliance rules

In order to express the BDCs, we propose the use of Numerical Constraints over Natural, Integer and Float domains in the following grammar. This is an adaptation of that proposed in [19], in addition relational database attributes have been added:

```
BusinessComplianceRules:=IDENTIFIER '←' Constraint
Constraint := Atomic_Constraint BOOL_OP Constraint
  |Atomic_Constraint | 'NOT'Constraint|'→'Constraint
BOOL_OP := 'AND'| 'OR'
Atomic_Constraint := function PREDICATE function
function := Variable FUNCTION_SYMBOL function
  |Variable|Constant
Variable := Table' · 'Attribute|Attribute|DataFlowVariable|Constant
PREDICATE := '=' | '<' | '≤' | '>' | '≥'
FUNCTION_SYMBOL := '+' | '−' | '*' | '/'
```

In general, a BDC is formed by a Boolean combination of numerical constraints. These numerical constraints can be specified by means of operators of comparison, attributes of databases, dataflow variables, and constants. This grammar provides the capacity of expressiveness of the BDCs, since BDCs constitute the formal representation of the relations between the items of data that form the business process. The limitations of use of the proposal appear when the constraints cannot be represented by numerical relations, data type, or by the operators included in this proposal, such as, when a relation between two variables is described by means of a trigonometric function. The limitation of the data domain and the operations that can be used are established by the solver employed in the Constraint Programming Problem, as explained in Section 5.3. Every commercial solver has the capacity to include the elements of our grammar, thereby making it possible to cover a significant number of problems.

The use of Constraints enables Integrity Rules, Derivation Rules, Reaction Rules and Production Rules to be represented. The rule can be of different types, depending on the instantiated and known variables. For example, for the constraint $hardwareCost + softwareCost = totalCost$:

- **If** *hardwareCost*, *softwareCost* and *totalCost* **are known**: The constraint represents an Integrity Business Rule.
- **If** *hardwareCost* **and** *softwareCost* **are known**: The constraint represents a Production Business Rule.
- **If** *hardwareCost* **and** *totalCost* **are known**: The constraint represents a Derivation or Reaction Business Rule, whereby *softwareCost* can be obtained.

The use of constraints permits the same business compliance rules to be reused, thereby avoiding the necessity of rewriting these rules in various locations of the process and for different uses. By using constraints and depending on the instantiation of the variables, it is possible to evaluate a tuple of values even when some values remain uninstantiated (stored in the database or introduced in the dataflow), which is equivalent to saying that the value of the variable is *null*. Hence, it enables the early detection and determination of errors, before all the values involved in the constraints are instantiated.

The use of '→' can help us to describe the if…then form, with the difference that '→' provides more information. For example, if A is unknown but B is true, we can assume that (A '→' B) is correct, since A '→' B is equivalent to $\neg A \vee B$, and therefore the possible values that satisfy B can be included in the analysis of correctness before A is instantiated. However, no deduction can be made on whether the {if A then B} structure is used.

It is therefore possible to validate BDCs that are not explicitly represented. For example, if the summation of hardware cost, software cost and human cost is equal to the total cost of the project, then the human cost is less than or equal to 10% of the software cost; and whether the summation of these three values is smaller than the total cost, then the human cost has to be less than or equal to 15% of the hardware cost. These BDCs can be expressed with the constraint: $(hardCost + softCost + humanCost = totalCost \wedge humanCost \leq hardCost * 0.10) \vee (hardCost + softCost + humanCost < totalCost \wedge humanCost \leq hardCost * 0.15)$ where hardCost[1..100], softCost[1..150], humanCost [1..100], totalCost[5..250] for the Float domain.

By using numerical constraints to represent BDCs, it is possible to infer other business rules, for example:

- $hardCost \leq totalCost$, $softCost \leq totalCost$, $humanCost \leq totalCost$,
- $humanCost \leq totalCost * 0.10$,
- if $hardCost = 10$ then $totalCost[12..161] \wedge human\ Cost = 1$.

3. Using business data constraints in business processes: a motivating example

In order to preserve the high data quality in business processes, a validation analysis is necessary. It is especially important in systems where there is a high degree of interchanged data, being necessary to maintain aspects such as soundness, completeness and correctness, according to the business goals. Through suitable checks, non-compliances due

to poor data quality problems can be detected and determined. The business processes where the data validation is more necessary are those that have a high level of human interaction, since humans can introduce intermittent faults in the system. The intermittence of the faults makes the detection and the diagnosis difficult, since an inconsistence detected while an activity is being executed, does not imply a malfunction in the activity or that this activity will fail in the future. Some typical examples where intermittent faults can be found are (i) Financial applications, where several data are introduced by hand in application forms; (ii) Medical applications that use data introduced by different types of medical staff and places, for example about blood test results, or Electronic equipments, which produce an important quantity of data, that can produce loose or corroded wire wrap, cracked solder joint or broken wire.

In this paper we have used an example of a real financial economic application to motivate the use of BDCs. The activities of the company are oriented towards negotiating collaborative projects between private companies and research groups as represented in Fig. 1. In the model, each task controls a state of the projects, where research groups, public and private companies work together. First of all, the R&D project is registered in the application, which including all the research groups, companies and the percentage of participation in each case. Depending on the project quality, it can be rejected or passed to the technical analysis, where the final quantity assigned to the project is in function of the external and internal evaluation of it. Derived from the accepted contract, various modifications can be included as addenda and contracts. When the project has finished, independent auditors or a branch auditor can be selected to evaluate the project. While the auditing is being executed, the economic justification is carried out. Finally, the activities related to the economic department are done: recover and payment, issue bills, etc.

All these tasks are carried out for a total of 25 employees belonging to 6 departments, which modify the stored information for more than 300 projects. Each employee is responsible for some activities of the process, between 5 and 20 R&D projects, depending on the department that they belong, and the month of the year. The persistence layer that supports this business process is formed by a database with 86 tables, 900 attributes within these tables, more than 112,200,000 tuples, 224 triggers and 107 integrity constraints. This implies that each employee can introduce an average of 200 data per project, during the 4 years that a project can take. The high quantity of introduced information, the number of people involved in each project, and the long duration of each instance execution produce a high degree of errors introduced in the database, which are detected too late. Summarizing, in these projects, an important work package was the validation of the input data of the users taking into account 270 Business Data Constraints with 435 variables involved. The use of BDCs facilitates enormously the consistency of the data in the information management. To introduce and maintain all the BDCs is a task for several employees who are business experts but not information technology experts, and hence it is necessary to introduce the BDCs into the system in a straightforward way. The BDCs are not described in a global way, since each task modifies certain data from a relational database and it is necessary to evaluate certain BDCs. Some examples of BDCs for some activities are presented below:

Execute application end:

project.idProject = idProject //here the dataflow variable idProject
//is used since in each instance only one project is modified

hardwareCost + softwareCost + humanCost = totalCost

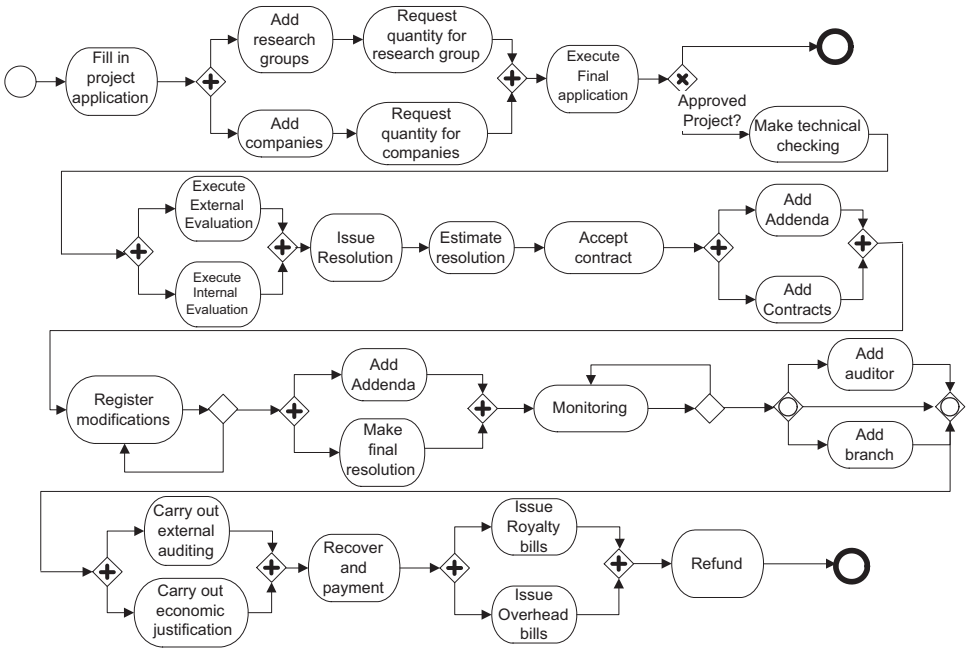

Fig. 1. Example of a real business process [18].

$softwareCost < humanCost$
$humanCost < 2 * hardwareCost$
Accept contract:
$project.idProject = idProject$
$totalCost + incentivePerYear = potentialIncentive$
$4 * softwareCost <= potentialIncentive$
Recover and payment:
$project.idProject = idProject$
$incentivePerCompany + humanCost > potentialIncentive$
$incentivePerCompany \geq hardwareCost + incentivePerYear$
$3 * softwareCost > incentivePerCompany$
$reducedQuantity \leq maxReducedQuantity$
$reducedQuantity \geq minReducedQuantity$

Since for each instance only one project is modified, the BDC: {$project.idProject = idProject$} could be described as an invariant of the whole process.

4. Extending the business process model for validation and diagnosis of business data constraints

In order to validate the BDCs, we are inspired by model-based diagnosis [12]. Model-based diagnosis consists of an analysis of the correctness of observations of a system (Observational Model) with the model of the system. The model of the business processes that we propose is formed by:

- A *Business Process Model* [53] formed by a set of activities, a set of control flow gateways (AND, OR, XOR) that describes the relationship between the activities, and a set of Data Objects that flows in the business process.
- A *Relational Database* whose data are involved in the business process model.
- The *BDCs* used in the validation of the compliance of the process that can involve data objects from the process or from the relational database. The domain of the variables of the Dataflow is defined by the designer in the business process model, and the domain of the stored variables is obtained from the database model. Each BDC can be associated with one activity, a set of activities or to the whole business process.

To perform the model-based validation and diagnosis of BDCs following the previous definitions, we propose a framework based on an extension of the classic Process Aware Information System (PAIS) framework [51]. A PAIS architecture [34] can be viewed as a 4-tier system, 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. In this paper, we have used the modification of the PAIS framework presented in [18] that adds a new layer for the validation of the BDCs (Audit Layer), as is shown in Fig. 2. This framework also modifies the location of the classic persistence layer that was originally accessible only from the Application Layer, and is now also accessible from the Audit Layer to facilitate database validation. The Audit layer is used from the Process layer, and depends on the business state or activity and the dataflow instances at each moment. To validate the BDCs, the relational model (Section 4.1) and the database where the BDCs are stored (Section 4.2) have to be analyzed. The dataflow and relational database function are the observational model, and BDCs are the model that must be satisfied.

4.1. Relational database model

Business processes can read and update data from databases. Therefore, data (the stored representation of facts in databases) is a fundamental component of information technology and business process management [49].

Business data is directly used in business operations and would be used even in the absence of computerized systems. The data model and the database are not the same thing, and the data model cannot simply be derived from the database by automated reverse engineering something that is often postulated as a solution where no data model exists. For instance, the database contains physical column names, but the BDCs will inevitably need the names of these columns and dataflow variables. Therefore, to combine both types of information, the relational database structure has to be known, and included in the validation process.

A *Relational Database* is a collection of predicates over a finite set of predicate 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. The set $\{A_1{:}D_1, A_2{:}D_2, \ldots, A_n{:}D_n\}$ is a

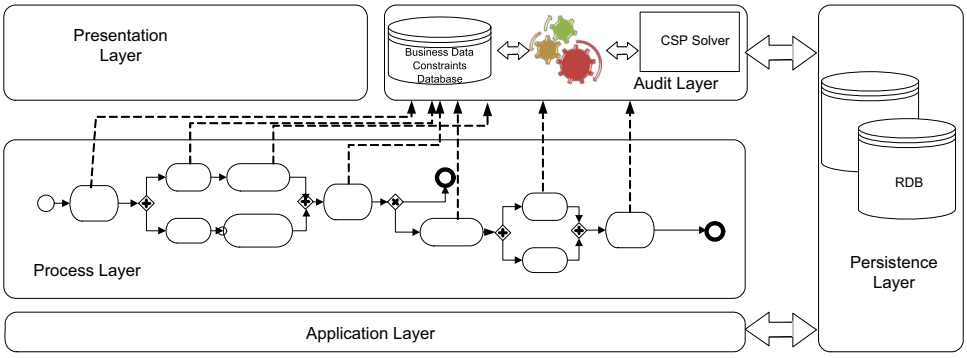

Fig. 2. Framework for run-time auditing [18].

relational-schema. A relation R defined over a relational-schema S is a set of assignments for each attribute for each domain. Therefore, the relation R is a set of n-tuples:

$\{A_1: d_1, A_2: d_2, \ldots, A_n: d_n\}$ where $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$.

Some of the attributes of a relation can be described as *Primary Key Attributes* which means that "two tuples of a relation cannot have the same values for their 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. Referential integrity is a database concept which ensures that relationships between tables remain consistent. When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record added to the linked table.

For the example of Section 3, a fragment of the relational model is shown in Fig. 3, where the information from each *Project* is related to each year in *ProjectPerYear*. At the same time, for each project and year, a set of *Companies* are involved by means of *ProjectPerYear&Company*. Depending on the Project, Year, and Company, a type of tax reduction is applied, limited by the table *Reduction*. An example of the values of the tuples for the relational database in a moment of one of the instances is presented in Fig. 4.

4.1.1. Representing the referential integrity as a graph

To extract the referential integrity between the tables, we propose to represent it by means of a graph. This graph represents all the tables and their relations by means of primary and foreign key references. To build the graph, the transformations are (i) for each table in the model, a node is created on the graph; for each relation between tables, a directed edge has to be created between the corresponding nodes on the graph, and (iii) the direction of the edge has to be, from the node of the table with the foreign key to the node of the table with the primary key. For the example of Fig. 3, the created graph is shown in Fig. 5.

4.2. Database of business data constraints

When a great deal of compliance rules has to be handled, the use of a database to store and manage these

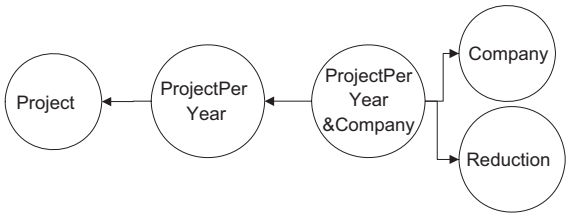

Fig. 5. Graph to represent the referential integrity of the relational model.

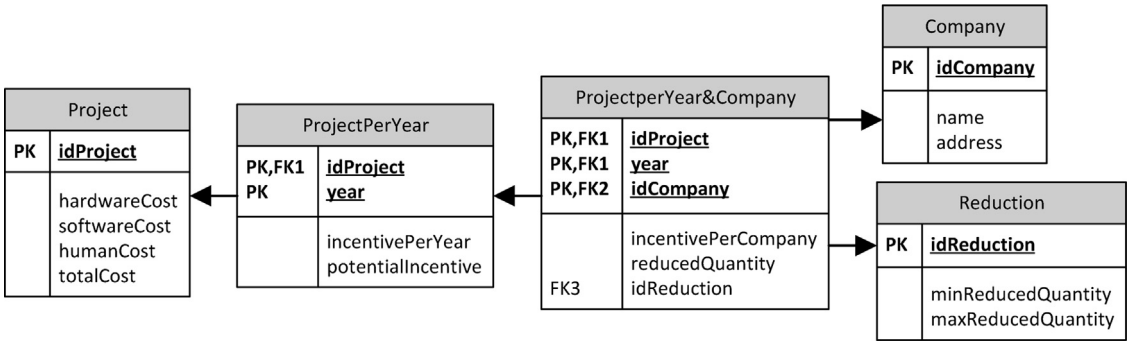

Fig. 3. Relational model for the example.

Project

idProject	hard Cost	soft Cost	human Cost	total Cost
223	2000	2250	3000	*null*
224	2000	1000	3000	6000
225	1948	2000	3000	6948
...	...	...	...	...

ProjectPerYear&Company

idProject	year	idCompany	incentivePer Company	reduced Quantity	idReduction
223	2009	501	*null*	15%	1
223	2010	501	*null*	*null*	1
223	2009	1040	5000	12%	null
224	2010	21	6800	8%	3
225	2008	225	4504	11%	4
...	...	...	...	...	...

ProjectPerYear

idProject	year	incentive PerYear	potential Incentive
223	2009	1000	*null*
223	2010	5500	7000
224	2008	1800	7948
...	...	...	...

Company

idCompany	name	address
501	H&Z	5th Av
1040	B&C	2th Av
21	T&H	6th Av
...	...	...

Reduction

idReduction	minReduced Quantity	maxReduced Quantity
1	5%	15%
2	6%	20%
3	5%	12%
...	...	...

Fig. 4. Example of tables of the relational database.

rules is a mandatory decision, especially when not all the compliance rules are established for the whole business process, and an indexation between activities and BDCs has to be defined. However, BDCs cannot be stored in a classic relational database, since storing a BDC also implies storing all the details related to its variables, domains, and data persistence relationships. The difficulty in storing BDCs is due to the problem of how to store numerical constraints that are not of a type supported by commercial databases. To manage Constraints, we propose the use of Constraint Database Management Systems (CDBMS) as explained in [27]. That proposal is based on an envelope over a database management system to manage Constraints as a classic type. This solution shields the user from unnecessary details on how the BDCs are stored and queried, how the three auxiliary tables (*Constraints*, *Variables* and *Constraints/Variables*) relate each constraint with its variables (Fig. 6). The table *Variables* stores the names of the variables, their identification and their type (Integer, Natural, or Float). In this paper, we enlarge the Constraint Database proposed in [27] to also include BDCs, where two new fields have been included in the table *Variables* (*Table* and *Field*) to keep the relation between metadata and the persistence data layer. This design enables the persistence layer design to be changed by only modifying the value in this table. These tables (Fig. 6) make it unnecessary to study all the constraints, when only the constraints with a set of variables need to be analyzed, for example to ascertain the BDCs related to the variable *incentivePerYear*, thereby reducing the evaluation time, as studied in [28].

Once how to store the BDCs is known, the next question is how each BDC is related to each activity. Fig. 7 represents the relations necessary to describe that a *Process* possesses a set of *Dataflow variables*, available for the *Activity* that formed this set. Each *BusinessDataConstraint* can be associated with a set of activities or to the whole process, while each Activity can have different associated BDCs. These associations are established in the table *Activity/BDC*.

5. Steps for the validation and diagnosis of business data constraints

Once all the parts that describe a business process model have been described, the validation and diagnosis process can be executed for each instance. As mentioned earlier, model-based validation can be performed in various activities to detect any inconsistency as early as possible. Therefore, during the execution of an instance, when an activity implies model-based validation, the model and the observational model have to be obtained. Hence, for each instance when an activity is audited, the following steps (depicted in Fig. 8) must be executed:

1. Obtain the BDCs related to the activity where the validation is being executed. This is a simple and efficient task thanks to the use of Constraint Databases as was explained in Section 4.2.
2. By using the obtained BDCs, the instance of dataflow variables, and the referential integrity graph, the tuples that represent the observational model are obtained, as detailed in Sections 5.1 and 5.2.
3. Instantiate the process model with the observational model, to discover the compliance of the BDCs. Since our proposal is based on representing the BDCs through numerical constraint, the evaluation of the compliance will be carried out by means of Constraint Satisfaction Problems (Sections 5.3 and 5.4).

5.1. Obtaining the observational model

In order to ascertain the tables and the tuples involved in the validation, we follow some of the ideas analyzed in [35], which are based on the referential integrity between tables. Before going into detail, it is necessary to explain and formalize why and how data stored in different tables is related in the validation and diagnosis process. These relations are derived from how the functional dependency is described by means of primary and foreign keys, keeping the relation between the tables.

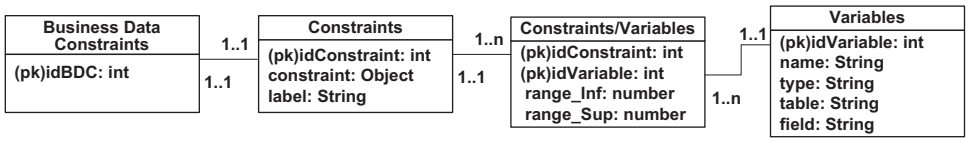

Fig. 6. Tables to store business data constraints [17].

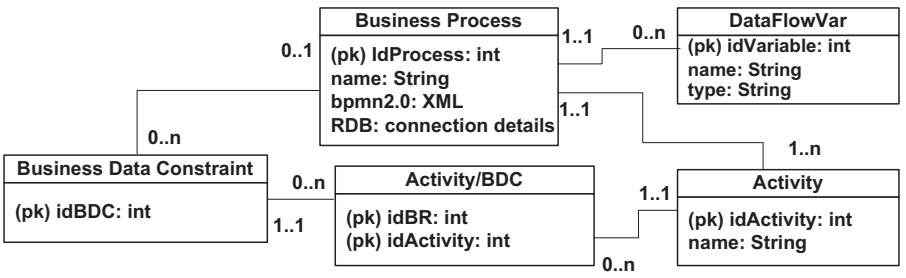

Fig. 7. Tables to relate BDCs and activities.

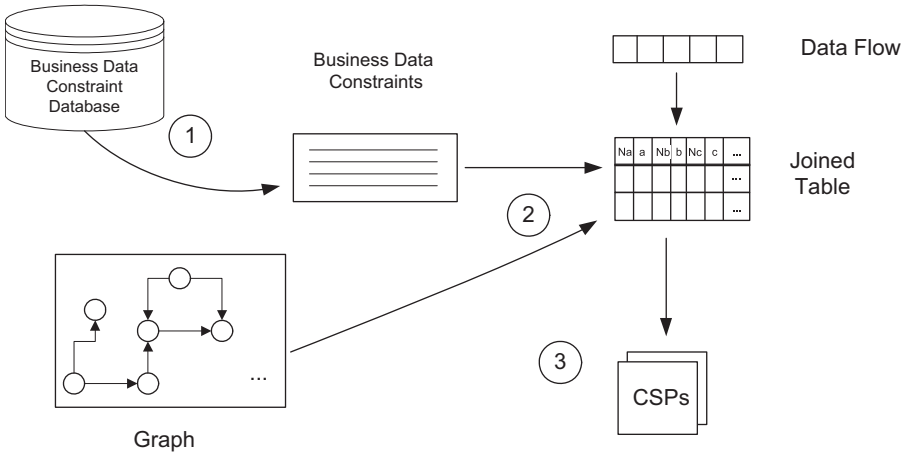

Fig. 8. Steps to validate the business data constraints.

Definition 1 (*Functional dependency*). A set of variables Y functionally depends on another set of variables X, represented by $X \rightarrow Y$, if, for a known and single value of X, there is also a known and single value of Y. In our example, for an IdProject value, only a single value of hardwareCost is possible, and hence idProject → hardwareCost.

Definition 2 (*Functional dependency graph*). This is a graph that represents the semantic context observed for a problem, where the nodes are the attributes of the relations and the edges represent the functional dependency.

In order to prevent any potential anomalies in relational models (such as manipulation, inserting, updating and deleting), the various Normal Forms proposed by Codd [11] present a way to divide the many attributes of the functional dependency graph into a number of tables, although they remain functionally and semantically connected. This functional dependency is maintained by means of the referential integrity (primary and foreign keys). Therefore, when attributes of various tables are involved in a BDC, this functional dependency has to be recovered.

The intensive representation of a relation R is the triple:

$$\langle \{A_1, \ldots, A_n\}, \quad PK = \{pk_1, \ldots, pk_i\}, \quad FK = \{fk_1(R_k), \ldots, fk_j(R_m)\}\rangle$$

where $\{A_1, \ldots, A_n\}$ are the attributes of R, $\{pk_1, \ldots, pk_i\} \subseteq \{A_1, \ldots, A_n\}$ are the primary keys, and $\{fk_1(R_k), \ldots, fk_j(R_m)\} \subseteq \{A_1, \ldots, A_n\}$ are the foreign keys, and where R_k, …, R_m are relations that have referential integrity with the relation R.

If t_1 and t_2 are the extensive representations of the relations R_1 and R_2 respectively, and

$t_1 = \{a_1, \ldots, a_i, \ldots, a_n\}$ where $a_i = fk(R_2)$ and $t_2 = \{a'_1, \ldots, a'_i, \ldots, a'_n\}$ where $a'_1 = pk$

Then t_1 and t_2 are semantically and functionally related, and could be in the same relation. To join these related attributes in a single tuple, the following operation can be used:

$t_1 \bowtie_{t1.fk = t2.pk} t_2$ which for the example becomes $t_1 \bowtie_{t1.a_i = t2.a'_1} t_2$

The join operation ($\bowtie$) obtains a new relation where the tuples of t_1 are concatenated horizontally with the tuples of t_2 whose primary key is equal to the foreign key of t_1.

Therefore, two items of data in different tables can be related by means of a functional dependency, which can be recovered if they are involved in the same BDC. Therefore, the steps to obtain the observational model are:

1. *Obtain the independent-variable clusters*. To reduce the complexity of the diagnosis process, we propose to determine the clusters of sets of BDCs with independent variables between them, and hence only the data involved with the BDCs in each moment is obtained. An independent-variable cluster is a set of BDCs whose variables are not involved directly or indirectly in another independent-variable cluster, formally expressed as:
   Definition: Cluster of BDCs with Independent Variables. Let BC represent all the BDCs of a process, let B be a set of BDCs where $B \subseteq BC$, and let $V(B)$ be the set of variables involved in B, then B is a Cluster of BDCs with Independent Variables iff $V(B) \cap V(BC - B) = \varnothing$, and B is minimal, which implies that $\nexists B'/B' \subset B | B'$ is a Cluster of BDCs with Independent Variables. The following steps will be executed for each independent-variable cluster in an independent way, since the data remains unrelated between each cluster. In our example: {incentivePerCompany + humanCost > potentialIncentive; incentivePerCompany ≥ hardwareCost + incentivePerYear; 3∗softwareCost > incentivePerCompany} and {reducedQuantity ≤ maxReducedQuantity; reducedQuantity ≥ minReducedQuantity}.
2. *Obtain the tables that contain any of the attributes involved in the BDCs*. These tables are the objective of our validation and diagnosis process, since their attributes have the observational model. For example, for the BDC {softwareCost > incentivePerCompany}, softwareCost and incentivePerCompany belong to the tables ProjectPerYear&Company and Project.
3. *Obtain all the related tables*. Sometimes, the tables obtained in the previous step are not the only tables involved in the process. In the previous example, since

the tables (Project and ProjectPerYear&Company) have no primary–foreign direct relation, it is necessary to determine the relation, in this case by means of the Table ProjectPerYear. Therefore, tables not involved in the BDCs sometimes appear in the validation process. With the tables obtained in the second step and by using the referential integrity graph, it is necessary to find the minimum spanning tree to include all the tables involved. For the example of the tables in Fig. 3, the minimum spanning trees for each independent-variable cluster are shown in Fig. 9.

4. *Build a relation with the involved data.* Once the minimum spanning trees are known, an expression for each tree is built to obtain all the necessary data stored for the BDCs validation. Each expression is a query which performs a join operation for all the tables involved, by following the relations in the calculated minimum spanning tree, thereby recovering all the fields from those tables. The form of the expression represented by relational algebra is:

Let $t_1, \ldots, t_n$ be the tables included in the minimum spanning tree
Let BDCs_Dataflow be the BDCs where dataflow variables are involved
Join_Table $= \sigma_{BDCs\_Dataflow}(\ldots \bowtie (t_i \bowtie_{ti.fk = tj.pk} t_j) \bowtie \ldots)$

where all the tables in $\{t_1, \ldots, t_n\}$ participate in the join conjunction (step 3), and the relation $t_i \bowtie_{ti.fk = tj.pk} t_j$ exists if there is an edge from t_j to t_i in the graph. The selection operator (σ) is included to obtain only the tuples involved in this instantiation, which implies the selection in terms of the instance values of the dataflow variables. This selection is feasible since it remains unnecessary to validate the correctness of any data that is not being modified in the instance, thereby reducing the computational cost. For one of the clusters of the BDCs of the activity "Recover and payment" in Section 3, and for the value of the dataflow variable idProject equal to 223, the expression would be $\sigma_{project.idProject = 223}$ ((ProjectPerYear&Company $\bowtie_{\ldots}$ projectPerYear) $\bowtie_{\ldots}$ Project), where all the join relations are in terms of the primary and foreign keys.

The relational algebra expression can be easily transformed into an SQL sentence, which could be evaluated for any relational database. For the example of the tables in Fig. 4, the result of this query is shown in Fig. 10. Each tuple represents an observational model that will be evaluated to ascertain the correctness of each BDC. At this point, additional consideration is required related to the various *null* values that appear. If we analyze the relation depicted in Fig. 10, where the non-instantiated values are represented with the value *null*, the values of the tuples can be related although they represent different observational models. For example, in the column called *potentialIncentive*, there are two *null* values, but in fact they represent the same value from the attribute *potentialIncentive* from the Table *projectPerYear*, for the primary keys {idProject: 223; year: 2009}. On the other hand, the two *null* values for the attribute *incentivePerCompany* represent different values since both come from the attribute *incentivePerCompany* of the Table *projectPerYearCompany*, but with different primary keys, {idProject: 223; year: 2009; idCompany: 501} and {idProject: 223; year: 2010; idCompany: 501} respectively. Therefore, the problem of extending the functional dependency remains for a later analysis of the possible values of *null* attributes, since it is unknown whether the values of the different tuples come from the same field. For this reason, it is necessary to "maintain" the relation before and after the tuples are joined to validate the correctness of BDCs. We propose the creation of a set of temporal tables where the name of the attributes concatenated to a sequence identifier are included as labels to the relations, thereby obtaining tables such as that shown in Fig. 11 for the table *projectPerYear&Company* in this example. The join relation that will be obtained is shown in Fig. 12, where the several values of *null* can be differentiated since they are associated with a label.

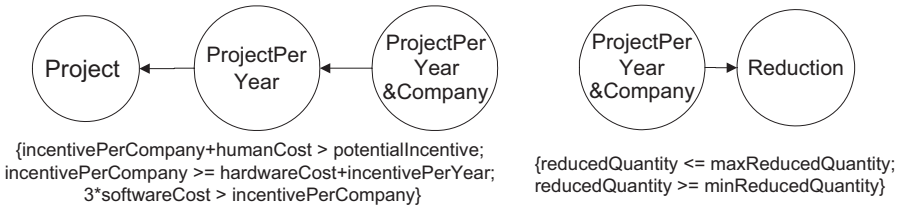

Fig. 9. Graph to represent the referential integrity of the relational model.

Joined Table									
idProject	year	idCompany	hard Cost	soft Cost	human Cost	total Cost	incentive PerYear	potential Incentive	incentive PerCompany
223	2009	501	2000	2250	3000	7250	1000	*null*	*null*
223	2010	501	2000	2250	3000	7250	5500	7000	*null*
223	2009	1040	2000	2250	3000	7250	1000	*null*	5000

Fig. 10. Relation with the tuples involved in terms of the referential integrity.

NamedProjectPerCompany						
idProject	year	idCompany	incentivePerCompany	namesIncPerCompany	reducedQuantity	nameReducedQuantity
223	2009	501	*null*	"IncPerCompany1"	15%	"ReducedQuantity1"
223	2010	501	null	"IncPerCompany2"	*null*	"ReducedQuantity2"
223	2009	1040	5000	"IncPerCompany3"	12%	"ReducedQuantity3"
224	2010	21	6800	"IncPerCompany4"	8%	"ReducedQuantity4"
225	2008	225	4504	"IncPerCompany5"	11%	"ReducedQuantity5"
...	...	...	...	...	...	...

Fig. 11. Table ProjectPerYearCompany with labels.

Joined Table with Names											
idProject	year	idCompany	...	total Cost	name totalCost	...	potential Incentive	name PotIncetive	incentive PerCompany	name IncPerCompany	...
223	2009	501	...	7250	"totalCost1"	...	*null*	*"PotIncentive1"*	*null*	"IncPerCompany1"	...
223	2010	501	...	7250	"totalCost1"	...	7000	*"PotIncentive2"*	*null*	"IncPerCompany2"	...
223	2009	1040	...	7250	"totalCost1"	...	*null*	*"PotIncentive1"*	5000	"IncPerCompany3"	...

Fig. 12. Joined relation with labelled tuples.

Left Joined Table with Names								
idProject	year	idCompany	reduced Quantity	nameReducedQuantity	name minRedQ	minReduced Quantity	name maxRedQ	maxReduced Quantity
223	2009	501	15%	"ReducedQuantity1"	*"minR1"*	5%	*"maxR1"*	15%
223	2010	501	*null*	"ReducedQuantity2"	*"minR2"*	5%	*"maxR2"*	15%
223	2009	1040	12%	"ReducedQuantity3"	*"minR3"*	null	*"maxR3"*	null

Fig. 13. Relation with labelled tuples involved in terms of the referential integrity.

5.2. Obtaining the observational model with null foreign keys

There is another problem derived from the case where any attribute is *null*. It is when a foreign key attribute is *null*. The referential integrity assures that if a foreign key has a value, it has to be one of the possible primary keys of the table referenced, but it is possible that the value is *null*. In this case, the values concatenated with the tuple to obtain the new relation should be *null*, since they are unknown at this moment of the evaluation. Then, to concatenate the tuples with null foreign keys with a null tuple, we propose a modification of the previous sentence using left join operator ($_{\bullet}\bowtie$) instead of join operator. R $_{\bullet}\bowtie_{R.fk = S.pk}$ S returns a new relation with all the tuples of R concatenated horizontally with the tuple of S with which primary keys are equal to the foreign keys of a tuple in R, or concatenated with a full null tuple if the foreign key is null.

Let $t_1, \ldots, t_n$ be the tables included in the minimum spanning tree
Let BDCs_Dataflow be the BDCs where dataflow variables are involved
Join_Table $= \sigma_{BDCs\_Dataflow}(\cdots {}_{\bullet}\bowtie (t_i {}_{\bullet}\bowtie_{tj.fk = ti.pk} t_j) {}_{\bullet}\bowtie \cdots)$

For example, we could consider the tuple of table *ProjectPerYear&Company*, where the attribute idReduction is *null* in the tuple whose primary keys are {idProject: 223; year: 2009; idCompany: 1040} (Fig. 4), and the obtained tuples would therefore be those shown in Fig. 13 by the sentence:

$$\sigma_{project.idProject\ =\ 223}((\text{ProjectPerYear\&Company} {}_{\bullet}\bowtie \text{ProjectPerYear}) {}_{\bullet}\bowtie \text{Project})$$

where all the join relations are in terms of the primary and foreign keys.

Once the involved data and BDCs are known, the question becomes how they can be combined to discover the unsatisfiable BDCs. We propose the use of Constraint Programming, since the definition of BDCs is very close to the definition of the logic and arithmetic constraint and, as mentioned earlier, it would be possible to analyze the possible values of the unknown variables (*null* variables) that appear in different BDCs. In the following section, the concepts necessary for the understanding of Constraint Programming paradigm are introduced.

5.3. Automating the run-time validation and diagnosis of BDCs in business processes

In order to ascertain whether the BDCs represented by Numerical Constraints are satisfied by the data related to the instance, we propose the use of Constraint Satisfaction Problems (CSPs). The CSPs represent a reasoning methodology consisting of the representation of a problem by means of variables, domains and constraints. Formally, it is defined as a triple $\langle X, D, C\rangle$ where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{d(x_1), (x_2), \ldots, d(x_n)\}$ 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 that simultaneously satisfy R. To solve a CSP, a combination of search and consistency techniques is commonly used [14]. The consistency techniques remove inconsistent values from the domains of the variables during or before the search. Several local consistency and optimization

techniques have been proposed as ways of improving the efficiency of search algorithms.

For a model-based validation process, we could model a CSP where:

- X is formed by all the *null* variables of the join table and by the dataflow variables that are not instantiated.
- D is defined for each variable depending on the type of the attributes for relational data, and the type of the dataflow variables.
- C is the set of BDCs involved with the activity that is evaluated in each instance whose relation is described in the table *Activity/BDC* (Fig. 7).

If there is a tuple of values for the variables X in the domain D, where all the BDCs of C are satisfiable, then the CSP solver will return a tuple with the possible values of X, and we can infer that the BDCs are satisfiable in this activity. The problem arises when there is no tuple of values for which the set C is satisfiable, and hence we would have no information about the possible BDCs that could be failing, only that the BDCs are unsatisfiable for the data involved in the instance. We propose determining the non-compliance subset of the BDCs by maximizing the number of BDCs that are satisfiable, since it is possible that not all the BDCs are satisfiable for one instance. In order to understand how we can model the CSP to determine this subset, two notions have to be introduced: Reified Constraints and Constraint Optimization Problems.

A reified constraint consists of a constraint associated with a Boolean variable which denotes its truth value. We will associate a Boolean variable to each BDC, for example $\{\text{Boolean } c_i = (\text{reducedQuantity} \leq \text{maxReducedQuantity})\}$.

This way of modelling the problems enables the variable c_i and $\{\text{reducedQuantity} \leq \text{maxReducedQuantity}\}$ to be unsatisfiable (false), and therefore the constraint $\{c_i = (\text{reducedQuantity} \leq \text{maxReducedQuantity})\}$ will be satisfiable (be true). However, the objective is to maximize the number of c_i variables that are true, and therefore an objective function is necessary. When an objective function f has to be optimized (maximized or minimized), then a Constraint Optimization Problem (COP) is used, which is a CSP with an objective function f. The maximal Constraint Satisfaction Problem (Max-CSP) consists of determining a total assignment which satisfies the maximum number of constraints. Max-CSP is an NP-hard problem and is generally more difficult to solve than the CSP problem. The basic method for solving this problem was designed by Freuder and Wallace [50]. Max-CSPs have already been used in model-based diagnosis [7], whereby alternative algorithms have also been proposed to improve the algorithmic determination of all minimal unsatisfiable subsets which use notions of independence of constraints and incremental constraint solvers [13] and structural analysis [16].

By using the constraint programming paradigm, it is unnecessary to wait until all the variables are instantiated to determine whether the BDCs are satisfied or not. We have also decided to use constraint programming since its advantages include: It is a very mature area that has been applied to very different problems related to optimization, and with high level of complexity; it uses propagation techniques to reduce the search space in an efficient way; there are numerous tools and algorithms to model and solve problems; it permits an easy definition of the BDCs using a wide range of constraints, such as implication constraints, disjunctive constraints, reified constraints, global constraints, and channelling constraints.

5.4. Improving the validation and diagnosis of the BDCs using clusters

Once the business process model, BDCs, dataflow variables, and the tuples of the stored data are known, the next question becomes how to combine all this information to determine the maximum set of BDCs that are satisfied. Since there are variables shared between the various tuples, and there are also different BDCs that involve the same variables, it is necessary to include in the same Max-CSP all the BDCs related by means of the variables in the database. Thereby the BDCs and variables will be involved in the same search for solutions. It would be possible to include all the BDCs and tuples in the same CSP, but it is desirable to create the Max-CSPs only with the variables and the related BDCs between them, to reduce the search area. This implies using the independent-variable clusters explained above. As we have mentioned, there are two independent-variable clusters in our example: $\{\text{incentivePerCompany} + \text{human Cost} > \text{potentialIncentive};\ \text{incentivePerCompany} \geq \text{hardwareCost} + \text{incentivePerYear};\ 3 * \text{softwareCost} > \text{incentivePerCompany}\}$ and $\{\text{reducedQuantity} \leq \text{max ReducedQuantity};\ \text{reducedQuantity} \geq \text{minReduced Quantity}\}$.

The breakdown of the problem into smaller ones, drastically reduces the number of possible combinations of BDCs in the determination of the minimum satisfiable BDCs. The worst case of combinations for n BDCs becomes 2^n, but if we determine the independent-variable clusters, the complexity of the combination is $2^{m_1} + 2^{m_2} + \cdots + 2^{m_k}$, where $m_1 + m_2 + \cdots + m_k = n$ and $2^{m_1} + 2^{m_2} + \cdots + 2^{m_k} \ll 2^n$. For the previous example, this will be: $2^3 + 2^2 < 2^5$.

Since each independent-variable cluster has no relation with the remaining clusters, an independent Max-CSP can be created and solved for each cluster. Each Max-CSP for any variable-independent cluster has the form:

type $\text{NullVar}_1, \ldots, \text{NullVar}_n$// the type depends on the attribute type
Boolean $\text{BDCs}_{11}, \ldots, \text{BDCs}_{1n}$
⋮
Boolean $\text{BDCs}_{m\ 1}, \ldots, \text{BDCs}_{mn}$
Integer varMaximize
BDCs_{11}=(Business_Rule$_1$ instantiated with the values of the tuple 1)
⋮
BDCs_{1n}=(Business_Rule$_1$ instantiated with the values of the tuple n)
⋮
$\text{BDCs}_{m\ 1}$=(Business_Rule$_m$ instantiated with the values of the tuple 1)
⋮
BDCs_{mn}=(Business_Rule$_m$ instantiated with the values of the tuple n)
$\text{varMaximize} = \text{BDCs}_{11} + \cdots + \text{BDCs}_{1n} + \cdots + \text{BDCs}_{m1} + \cdots + \text{BDCs}_{mn}$
maximize(varMaximize)

If any foreign key of the join table is *null*, then for each BDC_{ij} that has a relation with the attributes of a table that is not related, a constraint with the following form will be included:

$$\begin{aligned}(\text{name\_variable}_1 &= \text{value}_{1,1} \wedge \ldots \wedge \text{name\_variable}_n \\ &= \text{value}_{1,n}) \vee \ldots \vee (\text{name\_variable}_1 \\ &= \text{value}_{m,1} \wedge \ldots \wedge \text{name\_variable}_n \\ &= \text{value}_{m,n})\end{aligned}$$

where n is the number of attributes of the table referenced by means of referential integrity (Foreign and Primary keys) with a null value for the foreign key attribute, and where m is the number of tuples of the table referenced by means of referential integrity.

The Max-CSP created for the cluster {incentive PerCompany + humanCost > potentialIncentive; incentive PerCompany ≥ hardwareCost + incentivePerYear; 3 ∗soft wareCost > incentivePerCompany} presented above, with the data presented in Fig. 12 is:

Float potIncentive1, incPerCompany1, incPerCompany2
Boolean BDC_{11}, BDC_{12}, BDC_{13}
Boolean BDC_{21}, BDC_{22}, BDC_{23}
Boolean BDC_{31}, BDC_{32}, BDC_{33}
Integer varMaximize
//incentivePerCompany + humanCost > potentialIncentive
BDC_{11} = (IncPerCompany1 + 3000 > potIncentive1()
BDC_{12} = (`IncPerCompany2+3000>7000`)
BDC_{13} = (5000 + 3000 > potIncentive1)
// incentivePerCompany ≥ hardCost + incentivePerYear
BDC_{21} = (IncPerCompany1 ≥ 2000 + 1000)
BDC_{22} = (IncPerCompany2 ≥ 2000 + 5500)
BDC_{23} = (5000 ≥ 2000 + 1000)
//3∗softCost > incentivePerCompany
BDC_{31} = (3∗2250 > IncPerCompany1)
BDC_{32} = (3∗2250 > IncPerCompany2)
BDC_{33} = (3∗2250 > 5000)
varMaximize = BDC_{11} + BDC_{12} + BDC_{13} + BDC_{21} + BDC_{22} + BDC_{23} + BDC_{31} + BDC_{32} + BDC_{33};
maximize(varMaximize)

Model-based diagnosis is based on the parsimony principle [41]. It states that among competing hypotheses, the one with the fewest assumptions should be selected. For this reason we use Max-CSP which determines the minimum set of BDCs that cannot be satisfied. For the example, the Max-CSP finds that the minimum set of unsatisfied BDCs is $\{BDC_{22}\}$. The obtained BDCs from the Max-CSP involve a set of data, for the example if $\{BDC_{22}\}$ is incorrect, it does not mean that the constraint `incentivePerCompany ≤ hardCost + incentive PerYear` was incorrect, it means that some data related in the rule are incorrect producing an inconsistence in $\{BDC_{22}\}$.

It is also worth showing the Max-CSP created for the cluster {reducedQuantity ≤ maxReducedQuantity; reduced Quantity ≥ mixReducedQuantity} with the data presented in Fig. 13, since it includes foreign keys with *null* values:

Float reducedQuantity2, reducedQuantity3, minReducedQuantity3, maxReducedQuantity3
Boolean BDC_{41}, BDC_{42}, BDC_{43}
Boolean BDC_{51}, BDC_{52}, BDC_{53}
Integer varMaximize
//reducedQuantity ≥ minReducedQuantity
BDC_{41} = (15 ≥ 5)
BDC_{42} = (reduceQuantity2 ≥ 5)
BDC_{43} = (reduceQuantity4 ≥ minReducedQuantity4)
//reducedQuantity ≤ maxReducedQuantity
BDC_{51} = (15 ≤ 5)
BDC_{52} = (reduceQuantity2 ≤ 15)
BDC_{53} = (reduceQuantity3 ≤ maxReducedQuantity3)
((minReducedQuantity3 = 5 ∧ maxReducedQuantity3 = 15) ∨
(minReducedQuantity3 = 6 ∧ maxReducedQuantity3 = 20) ∨
(minReducedQuantity3 = 5 ∧ maxReducedQuantity3 = 12) ∨ …)
varMaximize = BDC_{41} + BDC_{42} + BDC_{43} + BDC_{51} + BDC_{52} + BDC_{53};
maximize(varMaximize)

The constraint {(minReducedQuantity 4 = 6 ∧ max Reduced Quantity4 = 20) ∨ (minReducedQuantity 4 = 5 ∧ maxReducedQuantity4 = 12) ∨ …} is not included in the maximization objective since it is not a Business Data Constraint itself that can be correct or incorrect. It is a constraint derived from the referential integrity used to restrict the possible values.

5.5. Evaluation complexity of Max-CSPs

Since the evaluation of the BDCs has been mapped onto a CSP, the validation time of the BDCs is linked to the complexity of the resolution of the CSP. This has been analyzed in great depth over recent decades [9], and depends on two parameters: the width of the graph and the order parameter. On one hand, the width of the graph represents the relation between the constraints, where the tractability in CSPs is due to the structure of the constraint network, where the tree-structured CSPs have polynomial complexity (linear with respect to the number of variables, and quadratic with respect to the cardinal of the domain of the variables). On the other hand, the order parameter, defined as the ratio of the number of forbidden tuples to the total number of possible combinations, determines the partition of the problems space into under-constrained, over-constrained and just-constrained problems. In the first two cases, the problems are scalable, but in just-constrained problems, a significant increase of solving cost could occur and the scalability would not be possible [61].

For these reasons, no affirmation about the efficiency or scalability in a generic way can be given by our proposal, since our framework permits any type or number of BDCs defined with numerical variables, and therefore the evaluation time will depend on the specific problem. Depending on the number of BDCs associated with each activity and the number of tuples, the Max-CSP will have more or less reified constraints and variables. The number of reified constraints, and therefore the number of variables that influence the variable to maximize is *the number of BDCs of each independent-variable cluster* multiplied by *the number of tuples*. However, whenever there is an incorrect BDC, the CSP created to validate the BDCs is over-constrained, and therefore is easy to solve.

In order to reduce the complexity and allowed the scalability of the problems, we propose:

- Include the detection of clusters to break the problem into smaller problems. Although there exist search strategies in the literature to find the clusters to

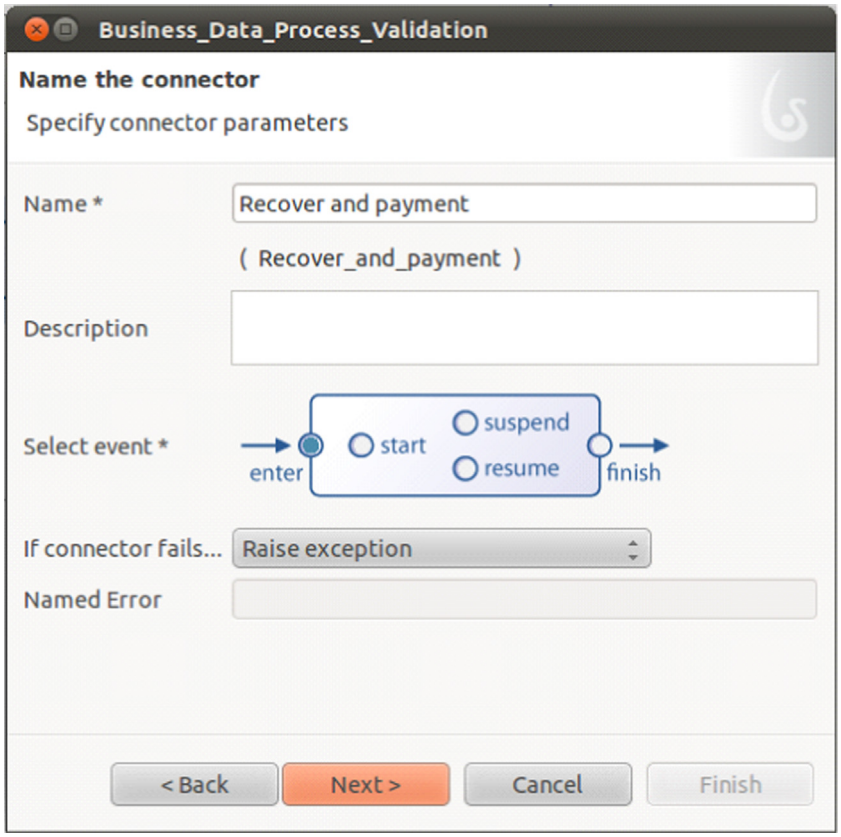

Fig. 14. Creating connector. Assignation of the connector the activity where the validation process is executed.

improve the CSP resolution [59,60], to the best of our knowledge they are not included in commercial tools, such as in Choco [38]. In addition, independent resolution of the clusters also helps to present the result of the validation. For example, if there are two clusters whose Max-CSPs determine that, for cluster 1, the BDCs that can fail are ($BDC_1 \vee BDC_2 \vee BDC_3$), and, for cluster 2, the possible incorrect BDCs are ($BDC_6 \vee BDC_7 \vee BDC_8$). If the BDCs of both clusters are included in the same Max-CSP, then the shown possibilities will be $(BDC_1 \wedge BDC_6) \vee (BDC_1 \wedge BDC_7) \vee (BDC_1 \wedge BDC_8) \vee \ldots \vee (BDC_3 \wedge BDC_6) \vee (BDC_3 \wedge BDC_7) \vee (BDC_3 \wedge BDC_8)$.
By implementing this cluster detection into our process, we enable the complexity to be drastically reduced, since the analysis of possible combinations is avoided, and the solution of the validation is more understandable than if all the possibilities were presented.

- Extract only the data involved in the instance from the relational database, thereby building simpler problems that are easier to solve. If it is only the data related to the instance (with the primary key *idProject* for the example) that can be modified, then they are the only data that participate in the validation process.
- Enable outsourcing of the validation process, since audit and business process layers can be executed in separate machines, as is recommended in rule evaluators, such as IBM in Drools [39].

6. Computational application for validation and diagnosis of BDCs

In order to make our proposal easy to utilize, two aspects must be considered: (1) ease of use for the process designer that has to model the business process and the BDCs, and (2) ease of adaptation for a process that already exists. In order to integrate our proposal into different types of applications and problems, we have implemented the audit layer engine in an independent application with an interface that can be used from any activity of any business process. Commercial tools permit a friendly interface to aid in the design of the business process model, but they lack the capacity to include data from the persistent layer and to evaluate the data even when some values remain uninstantiated. For this reason, we propose the implementation of the described framework by using a commercial tool where the module to create the BDCs, and the validation and diagnosis of data are included. To facilitate the creation of BDCs and the validation of the process, we have implemented an application and connectors that complement the framework presented in Section 4 with a set of technologies that could be replaced by other technologies. In order to show the steps for the configuration and use of our application, a video is available on http://www.lsi.us.es/quivir/mayte/validation.html. The steps for the design and execution of the validation and diagnosis process include:

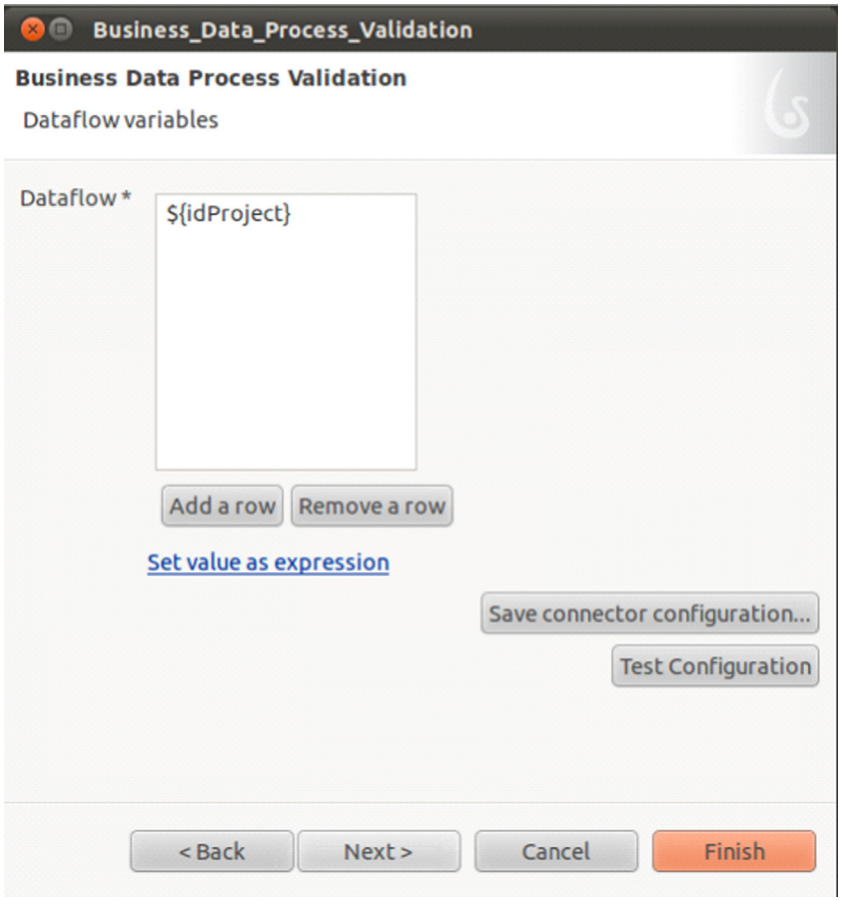

Fig. 15. Creating connector. Defining the dataflow variables that are involved in each validation.

1. *Modelling the business process.* The process can be modelled in any Business Process Management System, such as Intalio™, Activiti™, and Bonita Open Solution™. We have used Bonita Open Solution™ since it is an open-code application with a free distribution, commonly used in the private company sector.
2. *Locating the validations points*. Once the process is modelled, the designer must decide for which activities validation is desirable. If an activity participates in the BDCs validation, a connector must link the activity with the program that executes the validation. We have implemented a connector (Fig. 14), where the validation process can be executed at various instants of the execution. The connector, called "Business Data Process Validation", sends the dataflow values in run-time to the software that executes the steps explained in Section 5. Fig. 15 represents the part of the connector where the designer needs to define the dataflow variables involved in the activity. Once the minimal unsatisfiable set of BDCs is known, the connector returns the information about each variable to inform the user (Fig. 16).
3. *Creating the BDCs*. Once the process model is defined, it is necessary to create the BDCs and associated them to each activity. In order to store the BDCs and the relation between activities and BDCs, a CDB is used. The validation and diagnosis of the process uses the dataflow values obtained in run-time and the data obtained from the relational database, which is also included in this application. To this end, the implemented application facilitates the creation of BDCs for a relational model, since the application is configured with the information about the relational database and the XML of the process. When the XML of a process is analyzed, the tables *Business Process*, *Activities* and *DataflowVar* shown in Fig. 7 are automatically filled, since the XML of the process holds all this information. It is then possible to create and/or assign BDCs to the various activities. To provide a simple way for the business expert to add BDCs, the interface (shown in Fig. 17) permits many views of the relational database (UML, Relational Database or a textual description). These views help the designer to know the information stored in the tables of the relational database and the relation between them. Also it is possible to assign created BDCs to an activity (Fig. 18).
4. *Instantiating the Business Process Model.* Once all the parts of the process model have been defined, the execution process can be performed. When an instance arrives at an activity with a *Business Data Process* connector, the dataflow values of the instance and the identifier of the activity are sent to the software that

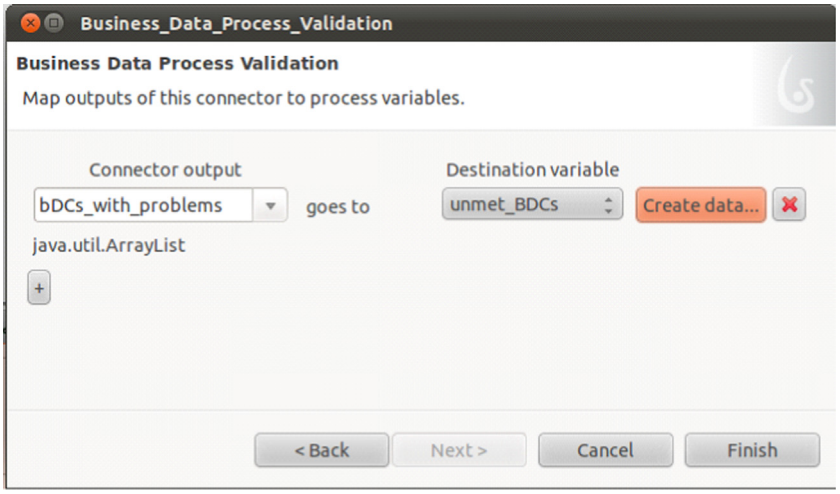

Fig. 16. Creating connector. Assignation of the diagnosis output (the minimal of unsatisfiable BDCs).

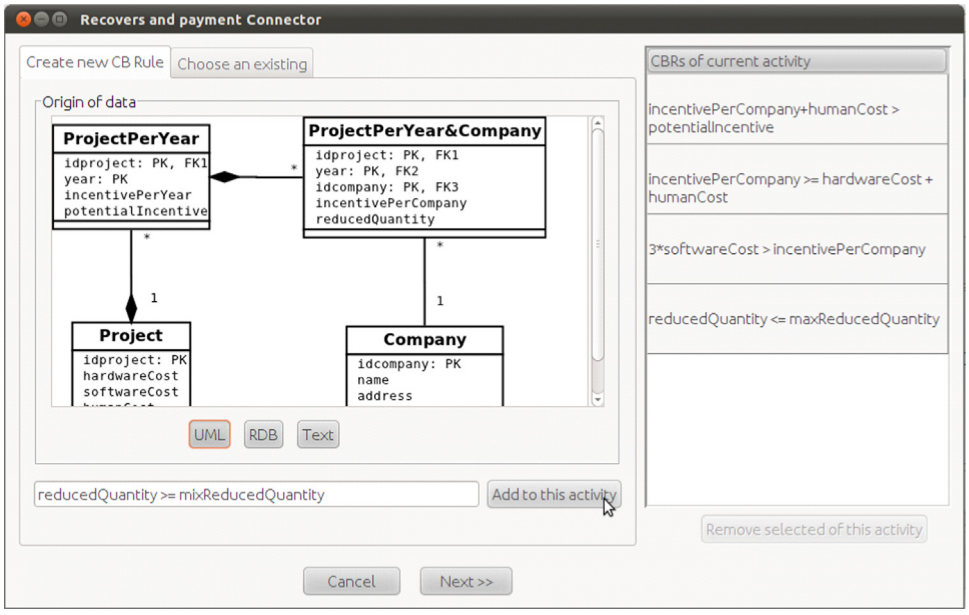

Fig. 17. Application to create a business data constraint.

implements the algorithm as explained in Section 5. This algorithm uses the BDCs related to each activity and the dataflow values to obtain the involved data stored in the relational database, and solve the Max-CSPs necessary to ascertain the unsatisfiable BDCs. The CSP solver used in our proposal is Choco [38].

7. Related works

Business Compliance rules can help to complete the process model information, since they can be used to validate business data. The importance of compliance data validation has been the focus of attention of numerous approaches, that can be classified into two types: a model-design analysis (compliance by design), and in a runtime analysis (runtime compliance check).

Compliance by design permits the analysis of dataflow to detect possible errors in design time, as in [46,45]. This analysis can be activity-centric, or artifact oriented. In the case of activity-centric orientation, a data-flow modeling approach tends to verify the data-flow in workflow systems. A variety of mechanisms have been developed to prevent errors at the structural level (e.g., deadlocks, livelocks, …) of the business process [47], even determining the degree of compliance [30], but they also have to comply with business level rules and policies. In [5], the activities are attributed with pre- and post-conditions that describe the data behaviour to verify the correctness of the model at design time. Artifact-centric orientation has been

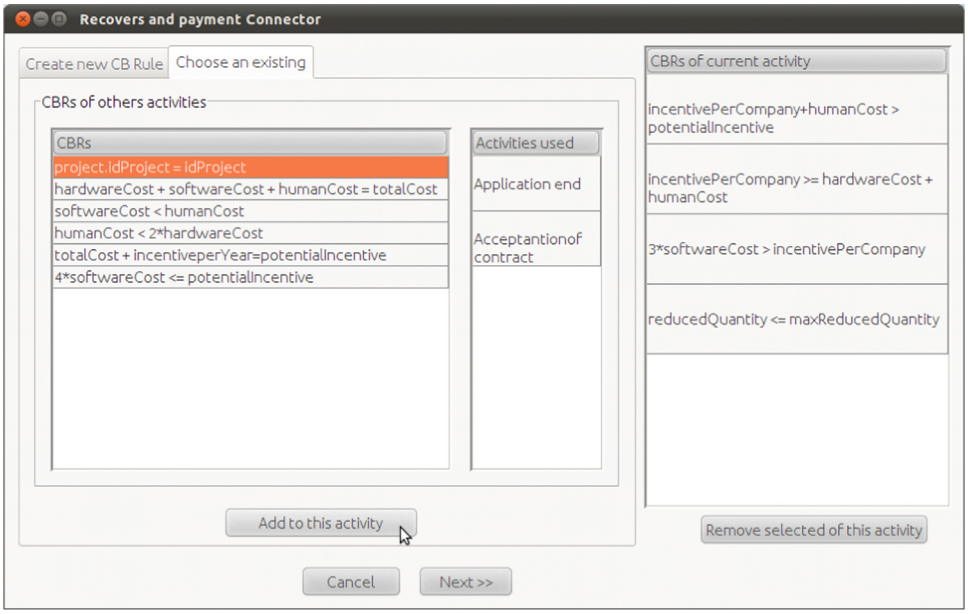

Fig. 18. Application to assign a business data constraint to an activity.

used to support consistent specifications [55], and some authors also include a set of operations that can be done for data: Initialize, Approve, Update, Read, Verify, Delete, that produce a set of anomalies: missing data, redundant data, conflict data. In [6], the authors describe a model-driven procedure to automatically transform an activity-centric model into a data-centric model of a BP that solves the limitations of other proposals: it deals with data anomalies in the original BP model that does not include information about the activities of the BP that are executed in the state transitions of the data object [24]. In the artifact-centric proposals, the authors are concerned about how the real data changes [26], and/or who changes it, but not about the values of the modified data for the behaviour described by compliance data. The main difference with our proposal is that other authors are unconcerned about the data values, only when they are read or modified.

A runtime compliance check permits the analysis of the instances of the business processes at runtime, and is complementary to compliance by design analysis. The solutions must enable frameworks to monitor of the systems to detect any non-compliance as soon as possible [57,18,29]. It can also be activity-centric oriented, or artifact-centric oriented. Most of the proposals are activity-centric oriented, and they analyze the compliance of the process model structure [43,31,10,58] or are related to model checking [25,1]. Related to how to model data-aware compliance rules, studies such as [25,52,32,3], have defined graphical notations to represent the relationship between data and compliance rules by means of data conditions. In [33], "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 [20], and an efficient algorithm for compliance verification using propagation is also discussed. In contrast to these approaches, artifact-centric proposals model the transformation of the data-object to be validated at instantiation time [42]. For example in [23], a preprocessing step to enable data-aware compliance checking in an efficient manner is presented.

Related to how the rules are integrated in a framework to complement the business process, in [56] there is an in-depth analysis about the integration of rules and process modelling and the shortcomings of the existing solutions. The framework of our work is based on [37], although we propose a separation of the evaluation into an independent layer that checks the compliance rules as a contract that describes the behaviour of the activities at different points of the business process instance, by taking into account the compliance rules evaluated in the future. In relation to the persistence layer and dataflow, relational databases have been used in the business process, for example in [48] which presents a solution where data are audited and stored on a relational database. However, no validation of the semantics is performed for this persistence layer and the business rules. In papers such as [2], the necessity to resolve the fault after detection is identified, unfortunately the data aspect is not included.

Summarizing, to the best of our knowledge, no solutions exist that validate and diagnose at runtime the data values of the persistence layer for the BDCs, where BDCs describe the behaviour of the data during a business process instance.

8. Conclusions and future work

In this paper, an innovative validation and diagnosis of BDCs has been presented, where various BDCs are associated with different activities in a business process. For an efficient management of BDCs, we have used Constraint Databases, Constraint programming and clustering analysis. These decisions have given rise to a framework that

shields the user from unnecessary details on how the BDCs are stored and validated. An important proposal of this work is how to extract the tuples involved by analysing the referential integrity of the relational database and how the automatic process of BDCs validation and diagnosis is modelled using Constraint Programming.

There are significant lines of research that can be analyzed in further depth, such as how can the grammar be modified to support other types of data (Set, Boolean, String, etc.); what actions can be taken when an inconsistency is determined; how it would be possible to automatically locate the rules in an effective way to improve the early detection of errors; and how the diagnosis can be modified to detect possible incorrect values.

Acknowledgement

The authors wish to thank Lesley Burridge for English comments. 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

[1] A. Awad, G. Decker, M. Weske, Efficient compliance checking using bpmn-q and temporal logic, in: BPM, 2008, 326–341.
[2] A. Awad, S. Smirnov, M. Weske, Towards resolving compliance violations in business process models, in: Proceedings of the Second International Workshop on Governance, Risk and Compliance—Applications in Information Systems, Amsterdam, The Netherlands, 2009.
[3] A. Awad, M. Weidlich, M. Weske, Visually specifying compliance rules and explaining their violations for business processes, J. Vis. Lang. Comput. 22 (1) (2011) 30–55.
[4] J. Becker, C. Ahrendt, A. Coners, B. Wei, A. Winkelmann, Modeling and analysis of business process compliance, in: M. Nüttgens, A. Gadatsch, K. Kautz, I. Schirmer, N. Blinn (Eds.), Governance and Sustainability in Information Systems, vol. 366, IFIP Publications, Springer, 2011, pp. 259–269.
[5] D. Borrego, R. Eshuis, M.T. Gómez-López, R.M. Gasca, Diagnosing correctness of semantic workflow models, Data Knowl. Eng. 87 (2013) 167–184.
[6] C. Cabanillas, M. Resinas, A.R. Cortés, A. Awad, Automatic generation of a data-centered view of business processes, in: CAiSE, 2011, pp. 352–366.
[7] R. Ceballos, R.M. Gasca, C.D. Valle, M. Toro, Max-csp approach for software diagnosis, in: IBERAMIA, 2002, pp. 172–181.
[8] S. Cetin, N.I. Altintas, R. Solmaz, Business rules segregation for dynamic process management with an aspect-oriented framework, in: Proceedings of the 2006 International Conference on Business Process Management Workshops, BPM'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 193–204.
[9] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, vol. 1, IJCAI'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 331–337.
[10] F. Chesani, P. Mello, M. Montali, F. Riguzzi, M. Sebastianis, S. Storari, Checking compliance of execution traces to business rules, in: Business Process Management Workshops, 2008, pp. 134–145.
[11] E.F. Codd, Relational database: a practical foundation for productivity, Commun. ACM 25 (2) (1982) 109–117.
[12] M. Cordier, F. Lévy, J. Montmain, L. Travé-massuyés, M. Dumas, M. Staroswiecki, P. Dague, A comparative analysis of ai and control theory approaches to model-based diagnosis, in: Fourteenth European Conference on Artificial Intelligence, 2000, pp. 136–140.
[13] M.G. de la Banda, P. J. Stuckey, J. Wazny, Finding all minimal unsatisfiable subsets, in: PPDP '03: Proceedings of the Fifth ACM SIGPLAN International conference on Principles and Practice of Declaritive Programming, ACM Press, 2003, pp. 32–43.
[14] R. Dechter, Constraint Processing. The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, 2003.
[15] M. El Kharbili, S. Stein, I. Markovic, E. Pulvermüller, Towards a framework for semantic business process compliance management, in: The Impact of Governance, Risk, and Compliance on Information Systems (GRCIS), CEUR Workshop Proceedings, vol. 339, Montpellier, France, 2008 1–15.
[16] R.M. Gasca, C.D. Valle, M.T. Gómez-López, R. Ceballos, NMUS: structural analysis for improving the derivation of all muses in overconstrained numeric csps, in: CAEPIA, 2007, pp. 160–169.
[17] M.T. Gómez-López, R.M. Gasca, Fault diagnosis in databases for business processes, in: Twenty-first International Workshop on Principles of Diagnosis, 2010, pp. 10–18.
[18] M.T. Gómez-López, R.M. Gasca, Run-time monitoring and auditing for business processes data using contraints, in: International Workshop on Business Process Intelligence, BPI 2010, Springer, 2010, pp. 15–25.
[19] M.T. Gómez-López, R.M. Gasca, L. Parody, D. Borrego, Constraint-driven approach to support input data decision-making in business process management systems, in: Information System Development, Springer, 2013, pp. 457–469.
[20] G. Governatori, J. Hoffmann, S.W. Sadiq, I. Weber, Detecting regulatory compliance for business process models through semantic annotations, in: Business Process Management Workshops, Lecture Notes in Business Information Processing, vol. 17, Springer, 2008, pp. 5–17.
[21] D. Hay, K.A. Healy, J. Hall, C. Bachman, J. Breal, J. Funk, J. Healy, D. Mcbride, R. Mckee, T. Moriarty, et al., Defining business rules. what are they really? the business rules group, Business (2000) 4–5.
[22] M.E. Kharbili, A.K.A. de Medeiros, S. Stein, W. van der Aalst. Business process compliance checking: current state and future challenges, in: P. Loos, M. Nüttgens, K. Turowski, D. Werth (Eds), MobIS, Lecture Notes in Informatics, vol. 141, 2008, pp. 107–113.
[23] D. Knuplesch, L.T. Ly, S. Rinderle-Ma, H. Pfeifer, P. Dadam, On enabling data-aware compliance checking of business process models, in: ER, 2010, pp. 332–346.
[24] J.M. Küster, K. Ryndina, H. Gall, Generation of business process models for object life cycle compliance, in: BPM, 2007, pp. 165–181.
[25] Y. Liu, S. Müller, K. Xu, A static compliance-checking framework for business process models, IBM Syst. J. 46 (2) (2007) 335–362.
[26] N. Lohmann, Compliance by design for artifact-centric business processes, in: Proceedings of the Ninth International Conference on Business Process Management, BPM'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 99–115.
[27] M.T. Gómez-López, R. Ceballos, R.M. Gasca, C.D. Valle, Developing a labelled object-relational constraint database architecture for the projection operator, Data Knowl. Eng. 68 (1) (2009) 146–172.
[28] M.T. Gómez-López, R.M. Gasca, Using constraint programming in selection operators for constraint databases, Expert Syst. Appl. 41 (15) (2014) 6773–6785.
[29] M.T. Gómez-López, R.M. Gasca, S. Rinderle-Ma, Explaining the incorrect temporal events during business process monitoring by means of compliance rules and model-based diagnosis, EDOC Workshops 23 (2013) 163–172.
[30] R. Lu, S.W. Sadiq, G. Governatori, Compliance aware business process design, in: Business Process Management Workshops, 2007, pp. 120–131.
[31] L.T. Ly, S. Rinderle, P. Dadam, Integration and verification of semantic constraints in adaptive process management systems, Data Knowl. Eng. 64 (1) (2008) 3–23.
[32] L.T. Ly, S. Rinderle-Ma, P. Dadam, Design and verification of instantiable compliance rule graphs in process-aware information systems, in: CAiSE, 2010, pp. 9–23.
[33] L.T. Ly, S. Rinderle-Ma, K. Göser, P. Dadam, On enabling integrated process compliance with semantic constraints in process management systems, Inf. Syst. Front. (2009) 1–25.
[34] H. Ma, Process-aware information systems: bridging people and software through process technology: Book reviews, J. Am. Soc. Inf. Sci. Technol. 58 (3) (2007) 455–456.
[35] M.J. Maher, D. Srivastava, Chasing constrained tuple-generating dependencies, in: ACM Symposium on Principles of Database Systems, ACM Press, 1996, pp. 128–138.
[36] D.C. McDermid, Integrated business process management: using state-based business rules to communicate between disparate stakeholders, in: Business Process Management, 2003, pp. 58–71.
[37] J. Meng, Achieving dynamic inter-organizational workflow management by integrating business processes, e-services, events, and rules (Ph.D. thesis), Gainesville, FL, USA, 2002 (chair-Su, Stanley Y. and Chair-Helal, Abdelsalam).

[38] G. Rochart, N. Jussien, X. Lorca, Choco: a java constraint programming library, Reference Manual. ⟨http://www.emn.fr/z-info/choco-solver/⟩.
[39] P. Browne, JBoss Drools Business Rules, Ed. Packt Publishing, 2009.
[40] R.G. Ross, Business Rule Concepts: Getting to the Point of Knowledge, Business Rule Solutions, LLC, 2013.
[41] Y. Peng, J.A. Reggia, Abductive Inference Models for Diagnostic Problem-Solving, Symbolic computation, Springer-Verlag, 1990.
[42] I. Rychkova, Exploring the alloy operational semantics for case management process modeling, in: Seventh IEEE International Conference on Research Challenges in Information Science (RCIS), Paris, France, 2013, pp. 345–356.
[43] S.W. Sadiq, M.E. Orlowska, W. Sadiq, Specification and validation of process constraints for flexible workflows, Inf. Syst. 30 (5) (2005) 349–378.
[44] N. Sponsor, Business rules and business processes, Inf. Syst. J. 1 (10) (2008) 20–24.
[45] S. Sun, L. Zhao, O. Sheng, Data flow modeling and verification in business process management, in: Americas Conference on Information Systems, 2004, pp. 4064–4073.
[46] S.X. Sun, J.L. Zhao, J.F. Nunamaker, O.R.L. Sheng, Formulating the data-flow perspective for business process management, Inf. Syst. Res. 17 (4) (2006) 374–391.
[47] N. Trcka, N. Sidorova, Data-flow anti-patterns: discovering data-flow errors in workflows, in: CAiSE 2009. Lecture Notes in Computer Science, vol. 5565, Springer, 2009, pp. 425–439.
[48] W. van der Aalst, K. van Hee, J.M. van der Werf, A. Kumar, M. Verdonk, Conceptual model for online auditing, Decis. Support Syst. 50 (3) (2011) 636–647.
[49] W. van der Aalst, A.H.M. ter Hofstede, M. Weske, Business process management: a survey, in: Business Process Management, 2003, pp. 1–12.
[50] R.J. Wallace, Directed arc consistency preprocessing, in: Constraint Processing, Selected Papers, Springer-Verlag, London, UK, 1995, pp. 121–137.
[51] B. Weber, S.W. Sadiq, M. Reichert, Beyond rigidity—dynamic process lifecycle support, Comput. Sci. R&D 23 (2) (2009) 47–65.
[52] I. Weber, J. Hoffmann, J. Mendling, Semantic business process validation, in: SBPM'08: Third International Workshop on Semantic Business Process Management at ESWC'08, 2008.
[53] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
[54] R. Wörzberger, T. Kurpick, T. Heer, Checking correctness and compliance of integrated process models, in: SYNASC, 2008, pp. 576–583.
[55] S. Yongchareon, C. Liu, X. Zhao, A framework for behavior-consistent specialization of artifact-centric business processes, in: Proceedings of the Tenth International Conference on Business Process Management, BPM'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 285–301.
[56] M. zur Muehlen, M. Indulska, Modeling languages for business processes and business rules: a representational analysis, Inf. Syst. 35 (4) (2010) 379–390.
[57] F.M. Maggi, M. Montali, M. Westergaard, W. van der Aalst, monitoring business constraints with linear temporal logic: an approach based on colored automata, in: S. Rinderle-Ma, F. Toumani, K. Wolf (Eds.), BPM 2011, Lecture Notes in Computer Science, vol. 6896, Springer, Heidelberg, 2011, pp. 132–147.
[58] Q. He, Detecting runtime business process compliance with artifact lifecycles. ICSOC Workshops, 2013, 426–432.
[59] P. Jégou, S. Ndojh Ndiaye, C.I. Terrioux, Computing and exploiting tree-decompositions for solving constraint networks, in: Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP-2005), 2005. pp. 777–781.
[60] X. Li, S.L. Epstein, Learning cluster-based structure to solve constraint satisfaction problems, Ann. Math. Artif. Intell. 60 (2010) 91–117.
[61] A. Krzysztof, Principles of Constraint Programming, Ed. Cambridge University Press, New York, NY, USA, 2003.