Fault diagnosis in databases for business processes

M. Teresa Gómez-López [1], **Rafael M. Gasca** [1]

[1] *University of Seville, Seville, Spain*
maytegomez@us.es
gasca@us.es

ABSTRACT

Business processes involve data that can be modified or updated by various activities. These data must satisfy the business rules associated to the process. As the information treated in a business process tends to be extensive, data are normally stored in a relational database, and hence the database has to be analyzed to determine whether the business rules are satisfied and what values are incorrect. This paper proposes the use of model-based diagnosis in the business processes scenario. This scenario combines business processes, business rules, relational databases and where the faults are the instances of the variables introduced by the users. These considerations make it necessary to introduce a new way for representing the model, and the design of new algorithms to solve it. This model provides a means for the detection of incorrect tuples of different tables of the database by avoiding the analysis of the full database. Furthermore, in order to manage the current business rules, the use of a constraint paradigm is proposed and by using Max-CSPs to isolate incorrect values.

1 INTRODUCTION

Organizations currently need to manage a great deal of data. This data must be conveniently gathered, transformed and stored according to a business data model. The evaluation of the correctness of data is crucial since none of the activities of a process can work correctly using incorrect values.

For the design of a whole business process management (van der Aalst *et al.*, 2003), it is necessary to design the database, the model of activities, and their causal and temporal relationships between them. Business rules can help to complete this information, since they can be used to validate business data (Chesani *et al.*, 2008). In a previous work (Borrego *et al.*, 2009) we have analyzed the faulty activities in business processes studying the choreography structural analysis, but where the database information is not taking into account.

The main idea is that the business process design and implementation are tested enough, hence the problem arises when some input values are introduced incorrectly by hand in the database, and these data can affect other business rules and data in future activities. There are papers (Guillou *et al.*, 2009) that consider data semantic faults, such as faults on input data, or database contents, or human faults. However no work is available which studies the fault diagnosis while taking into account the business rules and the relational database model. Auditing the stored information and dataflow is highly important since data are normally introduced by hand. Hence, this type of population of database produces numerous errors and inconsistent information which fluctuates. When a software activity works incorrectly, for the same input it will produce the same output, but this axiom is not true for human tasks. This paper takes an unashamedly data-oriented view of business rules engines, when there is a greater number of requirements, and a vast amount of data and rules. This renders it necessary to search for new solutions and to define higher expressiveness for business rules. Due to the complexity of business rules and data relations, it has become necessary to create a new way to represent, store, validate, and diagnose business rules depending on the data stored in a relational database. This paper is based on the validation of *Business Data Objects*, that are defined by the set of data stored in a relational database, which are updated in a business process instance. These Business Data Objects are changed for the different tasks of the process, passing through different *Business Object States*. When an unsatisfiability is found, the incorrect value has to be isolated. Based on these ideas, the proposals in this paper are:

- **To define a business rules language based on Constraints**. By using a Constraint Programming paradigm, the contract of business tasks can be represented at a more abstract level.
- **To redefine a repository to store business rules and relational database relations**. We propose the use of a Constraint Database to store the business rules and their relations with the relational

database model, necessary in a diagnosis process.

- **To propose a framework**. This framework is formed by a run-time auditing layer to check the conformity of the persistent data managed, and a diagnosis layer to isolate the tuples and the incorrect input data in a business process instance.
- **To diagnose whether there is a fault and what information of the relational database is incorrect**. The main idea is that since the business process design and implementation are tested enough, most of the problem comes from the input of incorrect values introduced by humans. We propose the use of Max-CSPs to find what values are incorrect for the business process model.

This paper is organized as follows: Section 2 presents the most interesting aspects related to Business Rules and the new orientation to *Constraints*. Section 3 analyses the fault diagnosis when the error is the input data and not the behaviour of the activities. Section 4 extends model-based diagnosis to business processes and relational databases. Section 5 sets out the proposed framework. Finally, conclusions and future work are presented.

2 BUSINESS RULES BY CONSTRAINTS

Business rules represent a natural step in the application of computer technology aimed at enhancing productivity in the workplace. When administrators of a business process want to change some functionality of the business, they have to wait for the reprogramming of system components. The adoption of business rules adds another tier to systems that automate business processes. Compared to traditional systems, this approach presents major advantages, as analyzed in depth in (Weber *et al.*, 2009), and includes: A lower cost incurred in the modification of business logic; a shorter development time; externalization of the rules and ease of sharing among multiple applications; faster changes with less risk.

If the expressiveness of business rules is improved, the above mentioned characteristics are also improved. For this reason, we propose the use of the *Constraints Paradigm* instance of the *if ...then ...* axiom that is used in current rules engines, such us Drools, Fair Isaac Blaze Advisor, ILOG JRules and Jess. *Constraints* proposed for the definition of business rules can be expressed as a Boolean combination with and/or operators of numerical equations and inequations for Integer, Natural and Float types.

The use of Constraints to represent business rules extends their formal semantics, since more knowledge can be represented and the description is less limited than when decision trees or a set of facts are employed. The use of Constraints enables Integrity Rules, Derivation Rules, Reaction Rules and Production Rules to be represented, and the evaluation of whether a set of data is correct for a company policy. For example, if it is necessary to check that the summation of hardware cost, software cost and human cost is equal to the total cost of the project, and that the human cost is smaller than 10% of the software cost, and that when the summation of these three values is smaller than the total cost, then the human cost has to be smaller than 15% of the hardware cost. These business rules can be expressed with the constraints: $(hardCost + softCost + humanCost = totalCost \wedge humanCost \leq hardCost * 0.10) \vee (hardCost + softCost + humanCost < totalCost \wedge humanCost \leq hardCost * 0.15)$ where hardCost$[1..100]$, softCost$[1..150]$, humanCost$[1..100]$, totalCost$[5..250]$ for Float domain.

By using Constraints to represent business rules, it is possible to validate knowledge that has to been explicitly described about stored variables in the database. Some examples of the inferred business rules for the above constraints can be:

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

The knowledge that is represented by constraints is wider than the current business rules languages can describe. By using Constraints and depending on the instantiation of the variables, it is possible to evaluate a tuple even if some variables are not instantiated (stored in the database). Hence, it permits an early detection of faults before the full tuple of values of variables is fixed. In order to infer these unknown values, a Constraint Satisfaction Problem (CSP) can be created.

The CSPs represent a reasoning framework consisting of variables, domains and constraints. Formally, it is defined as a triple <*X, D, C*> where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \ldots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \ldots, C_m\}$ is a set of constraints. A constraint $C_i = (V_i, R_i)$ specifies the possible values of the variables in V simultaneously in order to satisfy R (Dechter, 2003).

By using the Constraint paradigm, when the values of the variables related to a business rule are determined in various tasks of a business process, it is not even necessary to wait until all the variables are instantiated to determine whether the business rules are satisfiable.

3 CONSTRAINT PROGRAMMING TO FIND AND ISOLATE INCORRECT VALUES OF VARIABLES

As explained in the previous section, a CSP consists of assigning values to variables which are subject to a set of constraints. A solution of a CSP is a total assignment satisfying every constraints. When the CSP has no solutions, we are interested in finding an assignment which satisfies as many constraints as possible. The maximal constraint satisfaction problem (Max-CSP) consists of finding a total assignment which satisfies the maximum number of constraints. Max-CSP is an NP-hard problem and generally is more difficult to solve than the CSP problem. The basic complete method for solving this problem was designed by Freuder and Wallace (Wallace, 1995). Max-CSPs have already been used in model-based diagnosis (Ceballos *et al.*, 2002), although other different algorithms has

also been proposed in order to improve the algorithmic determination of all minimal unsatisfiable subsets using notions of independence of constraints and incremental constraint solvers (de la Banda *et al.*, 2003) and structural analysis (Gasca *et al.*, 2007).

Most times the problem is not related to incorrect restrictions nor a malfunction of the activities, very often the problem is due to incorrect input data. If we want to model a business process as a set of values for a set of business rules, where the error is the instance of a variable for a tuple in a database, the constraints that can be relaxed to obtain a correct behavior will be the values of the variables. It means, to find the minimum number of variables that whether they are not instantiated, all the constraints are satisfiable.

Not all the variables can be incorrect for a direct human task, since some of them only depend on the values of input variables. For example, for the constraint $c = a + b$, a and b are introduced by an user and c is derived from the values of a and b. It implies that if a or b are incorrect, them c will be incorrect too. For this situation, two types of variables are defined:

Definition 1: Input Variable. Variable whose value is introduced by the user in an activity of a business process.

Definition 2: Derived Variable. Variable whose value depends on the value of other variables (input or derived variables). It means that its value is not introduced as an input variable.

Once the input and derived variables are determined, depending on the business process, the goal of a diagnosis process is to detect the possible minimal incorrect variables set.

Definition 3: Possible Minimal Incorrect Variables Set. Set of input variables (S) that if they are not instantiated, the set of business rules are satisfiable. This set is minimal iff there is not a subset SS $\subset$ S where SS is a possible minimal incorrect variables Set.

The Max-CSP that is created to obtain this minimum set of variables is:

type InputVar_1, . . ., InputVar_n
type InputVarBool_1, . . ., InputVarBool_n
type DerivedVar_1, . . ., DerivedVar_m
type varMaximize
InputVar_1 = [min_domain_1, max_domain_1]
. . .
InputVar_n = [min_domain_n, max_domain_n]
DerivedVar_1 = [$\text{min\_domain\_derived}_1$, $\text{max\_domain\_derived}_1$]
. . .
DerivedVar_m = [$\text{min\_domain\_derived}_m$, $\text{max\_domain\_derived}_m$]
Business_Rule_1
. . .
Business_Rule_k
Constraint_1 = (InputVar_1=[InputVar_1_Instance])
. . .
Constraint_n = (InputVar_n=[InputVar_n_Instance])
varMaximize = Constraint_1 + . . . + Constraint_n
maximize(varMaximize)

3.1 Diagnosis example for incorrect input data

Figure 1 depicts a business process where the values of some variables are introduced by the users in different tasks. Different business rules can be evaluated in different moments of a process instance. For this example, the two first sets of business rules are satisfiable, but the third set is not satisfiable for the values shown in the figure. For the example, the input variables are {softwareCost, hardwareCost, humanCost, incentivePerYear, incentivePerCompany}, and the derived variables are {totalCost, potentialIncentive}. If a Max-CSP is created in the way presented in the previous section, it is found that if the value of *SoftwareCost* is not instantiated as 1000, all the business rules will be satisfiable. Hence, the Possible Minimal Incorrect Variables Set is {*SoftwareCost*}.

4 MODEL-BASED DIAGNOSIS FOR DATABASES IN BUSINESS PROCESS

In order to describe a business process where the activities modify a set of values of a database, it is necessary to extend the definition of model for model-based diagnosis. Comparing the problem with the classic diagnosis, the activities of a business process work such as components where the number of variables (inputs and outputs) is unknown since in an activity several tuples of several tables can be modified. We have decided to use the relational database theory to relate the information stored in the database with the business rules that work as a contract or a model of the system.

Most computer applications read and update data from databases. Therefore, data (the stored representation of facts in databases) is a fundamental component of information technology. Improvements in the integration of data in business processes are necessary, since it is common that not all information is transferred by means of dataflow, but is modified via a database.

Business data is data that is directly used in business operations and would be used even in the absence of computerized systems. Metadata is additional data that describes what these computerized systems contain and how they work, or describes the business data, such as definitions of business terms. In order to define the equivalence between the business rules and data persistence layer, the BOM (Business Object Model) was introduced (Heumann, 2001), although the relation between persistence layer and business layer was not defined. Current architectures contain no dataflow integrity and audit trail since all business logics are hard-coded. This means that business processes cannot be easily related to any which involves complete dataflow traceability.

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 only physical column names, but the rules engine will inevitably need the names of these columns. Hence, each business rule has to be transformed into a query evaluation over real tables and columns with

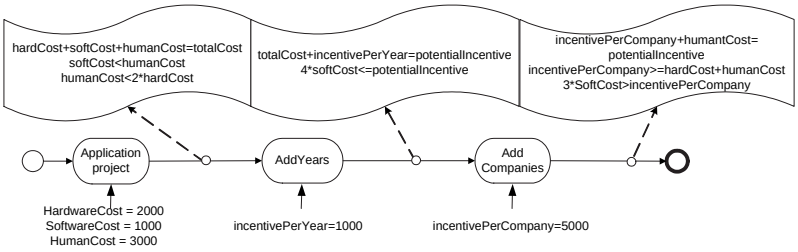

Figure 1: Example of a business process with input and derived variables

a condition. The relation between business rules variables and database fields will be stored in a Constraint Databases in order to establish the relations between variables, dataflow and database fields.

4.1 Model of the problem

The model of a business process for the diagnosis of a Database is formed by:

- **Relational Database:** It is a collection of predicates over a finite set of predicate variables described by 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. Being A_1, A_2, ..., A_n attributes for 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 relational-schema. A relation R defined over a relational-schema S is a set of assignments to each attribute to each domain. Then, 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* that means that "two tuples of a relation cannot have the same values for their primary key attributes".
  Two tables can be related by means of their primary and foreign keys variables described in the bibliography as the relational model.
  For the example presented in Figure 1, the relational model is shown in Figure 2.

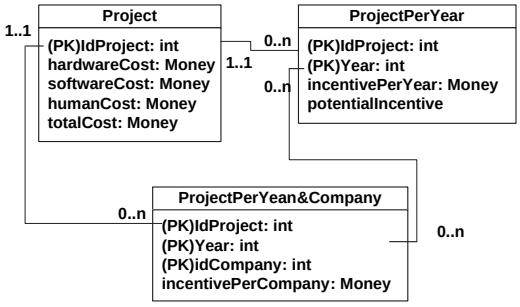

Figure 2: Relational Model for the Example

- **Observational Model:** It is the tuples that form the database that are monitored and diagnosed. Each attribute of the tuple can be an *Input Variable*, *Derived Variable*, *Key Variable* or *Query Variable*. Input and Derived variables have already been defined, but once the relational model has been introduced, the definition of *Key Variables* and *Query Variables* must be included.

  Definition 4: Key variables. Set of variables that differentiate a tuple from another. They correspond to the primary key attributes of the relational model. These variables will form a very important part in the diagnosis process since they would permit to isolate a tuple from another.

  Definition 5: Query variables. Set of input variables and subset of key variables used to describe the set of tuples introduced and/or modified in an activity. These variables depend on the activity, and it depends on the moment when the diagnosis process is executed. These variables can be transferred through the activities as variables of the dataflow when they do not change for a process instance. For the example, *ProjectNum* represents the number of the project related to the hardCost, humanCost and softCost for the first activity; *ProjectNum* and *Year* represent the project and the year related to the incentivePerYear for the second activity; and *ProjectNum*, *Year* and *Company* represent the incentivePerCompany for the third activity. As the number of the project does not change for a process instance, it is only introduced in the first activity, and the same happens with *year* in the second activity.

  Being a tuple a set of attributes $\{A_1,\ A_2,\ \ldots,\ A_n\}$, Input Variables (IV), Derived Variables (DV), Key Variables (KV) and Query Variables (QV), the properties of these set of variables are:
  - $\{A_1, A_2, \ldots, A_n\} = \text{IV} \cup \text{DV} \cup \text{KV} \cup \text{QV}$
  - $\text{QV} \subseteq \text{KV}$
  - $\text{QV} \subseteq \text{IV}$
  - $\text{IV} \cap \text{DV} = \emptyset$

  When the business rules are evaluated, only the tuples that are described by these keys are selected and included in the evaluation. The main problems are to describe in which fields are the stored variables (input and derived); which are the keys that can differentiate a tuple from another; and how the variables of different activities evaluated for different business rules are related.

- **Business process, business rules and relation between rules and database fields:**
  The diagnosis process has to take into account: the activities that form the business process; the

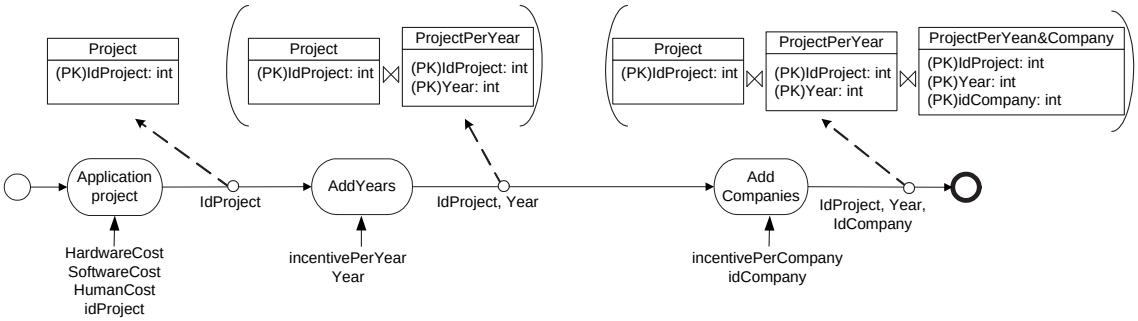

Figure 3: Business process with tables for the example

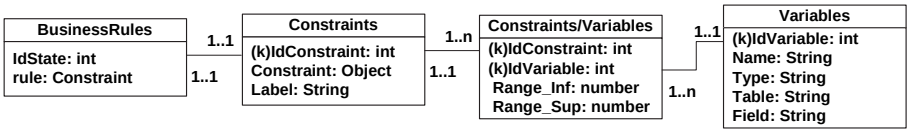

Figure 4: CDB tables to index business rules with constraints, variables, tables and fields

business rules that are involved; and the relation between the variables and the database fields, as it is shown in Figure 3 by using the same business rules presented in Figure 1. When a great deal of business rules have to be handled, the use of a database is a mandatory decision (Chisholm, 2003), especially when not all the business rules are established for the whole business process. The storage of business rules also implies storing all the details related to its variables, the domain of the variables and data persistence relationships. Also, it implies to store the correspondence between the variables of the business rules, the fields and tables in the database, and the relational model. These types of information and business rules expressed by Constraints are supported by Constraint Database Management Systems (CDBMS).

Constraint Databases (CDBs) were initially developed in 1990 with a paper by Kanellakis, Kuper and Revesz (P. C. Kanellakis and Revesz, 1990). The basic idea behind the CDB model is to generalize the notion of a tuple in a relational database to a conjunction of constraints, since a tuple in relational algebra can be represented as an equality constraint between an attribute of the database and a constant.

We propose to use the architecture presented in (Gómez-López *et al.*, 2009). It stores numerical constraints as objects indexed by the variables contained within, hence, when a CDB is created, three auxiliary tables are also automatically created (*Constraints*, *Variables* and *Constraints/Variables*) which relate each constraint with its variables (Figure 4). The table *Variables* stores the names of the variables, their identification and their type (Integer, Natural or Float), and for business rules, two new fields have been included in the *Variables* table (*Table* and *Field*) to store the relation between metadata and persistence data layer.

The use of CDB allows the business rules (represented by Constraints) can be associated to different moments of business process, in order to avoid the evaluation of all business rules, and the full database. Since not all the business rules are activated in the whole business process (McDermid, 2003).

4.2 Diagnosis Process

In order to diagnose the input variables of a database for a business process instance, it is necessary to extend the diagnosis process developed in Section 3 for several tuples. The first problem is to determine which tables and attributes are involved, and after that the tuples that will be evaluated. The steps are: (a) To select the tuples; (b) To analyze the satisfiability of each tuple; (c) If a tuple is not satisfiable, a Max-CSP will be created for the diagnosis; (d) To inform about the error and the fault instances. The Figure 5 shows the steps in a graphical way.

To select the tuples to be evaluated by the business rules

In order to select the involved tuples, the business rules stored in the CDB and the query variables of the dataflow (Step 1 of Figure 5) will be combined to form a selection operation (Step 2 of Figure 5). The variables of the business rules are indexing by means of the CDB with the attributes of the tables of the relational database. It means that all these tables will participate in the query, doing the join operation over these tables having in account the relation between primary and foreign keys represented in the relational database model. The queries have the form:

```
SELECT list_Of_Attributes_to_Obtain
FROM list_tables WHERE condition.
```

The parameters in this case are:

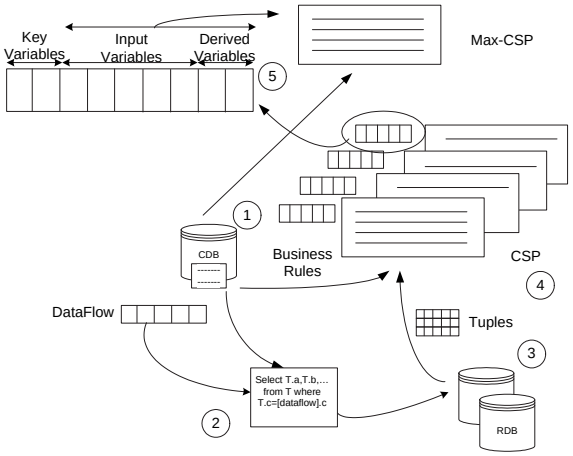

Figure 5: Audit Layer Procedure

- **list_Of_Attributes_to_Obtain**: Input Variables ∪ Derived Variables ∪ Key Variables
- **list_Of_Tables:** ⋃ Variables.Table of the CDB (Figure 4) where Variables.name ∈ {list_Of_Attributes_to_Obtain}
- **conditions:** t_i.primaryKey=t_j.primaryKey AND ... AND t_k.queryVariable=[instanceValue] AND .... Where $\{t_i, t_j, t_k, \ldots\}$ =list_Of_Tables and the relation between the primary keys of the different tables is determined in the relational model (Figure 2).

For the example of Figure 3 the selection operator is:

```
SELECT
    P.idProject, PY.year, PC.idCompany,
P.softCost, P.hardCot, P.humanCost, P.totalCost,
PY.incentivePerYear, PY.potentialIncentive,
PC.incentivePerCompany
FROM
   Project as P, ProjectPerYear as PY,
   ProjectPerCompany as PC
WHERE
   P.idProject=PY.idProject AND
   PY.idProject=PC.idProject AND
   PY.year=PC.year AND
   P.idProject = 223 AND
   PY.year = 2009 AND PC.idCompany = 501;
```

To analyse the satisfiability of each tuple

Once the tuple or tuples are known (Step 3 of Figure 5), a CSP will be build only with the input and derived variables (Step 4 of Figure 5). The key variables are not necessary in the CSP since they are only needed to isolate a tuple from another, and these variables do not have to be involved mandatory in the business rules.

To diagnose each unsatisfiable tuple

If a tuple is unsatisfiable, a Max-CSP is created for each faulty tuple, including all the business rules of the path covered for the process instance (Step 5 of Figure 5).

To inform about the faulty instances of the variables

The business process diagnosis that we propose obtains the minimal set of fields of the tables for an only one tuple whose values are not correct for the business rules. In order to isolate the minimal incorrect variables, also it is important inform about the key variables that ensure that this tuple can be differentiated from the rest of the tuples of the database.

The tables of the problem and the resulted tuples for the query sentence are presented in Figure 6, with three examples of output tuples. For the *tuple A*, all the business rules are satisfiable, and the diagnosis is not necessary. For the *tuple B*, the business rules are unsatisfiable, and a Max-CSP is created with all the business rules. The instance error is found in the variable *SoftCost*, that belongs to the table *Project* which primary key is *idProject*, it implies that the result of the diagnosis will return: {idProject: 224 faults in softCost: 1000}. The *tuple C* is not satisfiable, and the diagnosis process finds that the minimal incorrect variables are *hardCost* and *IncentivePerCompany*, where the primary keys of both are *idProject*, *year* and *idCompany*, hence the diagnosis will return: {idProject: 225, year: 2008, idCompany: 225 fault in hardCost: 1948, incentivePerCompany: 4504}.

5 RUN-TIME AUDITING AND DIAGNOSIS FRAMEWORK

In order to permit the validation of business data in different moments of the business processes execution (states), and to represent and store business rules using the Constraint Programming paradigm, we propose an extension of the classic Process Aware Information System (PAIS) framework. This framework enables the tasks to be audited according to their associated set of business rules, the dataflow to be monitored and evaluated for the business rules.

Increasingly, business rules are also viewed as a critical component of Business Process Management solutions, due to the need to ensure flexibility. Some analysts believe the combination of business rules technology with Business Process Management offers an agile approach to workflow and enterprise integration. The definition of an auditor of business data objects into separated layers enables the updating of processes or rules.

In this context, the notion of PAIS provides a guiding framework to understand and deliberate on the above developments (Ma, 2007), (Weske, 2007). In general, a PAIS architecture can be viewed as a 4-tier system as presented in (Weber *et al.*, 2009), where from top to bottom the layers are: Presentation Layer, Process Layer, Application Layer and Persistency Layer. As a fundamental characteristic, a PAIS provides the means to separate process logic from application code. We propose a new framework shown in Figure 7, where two new layers are added to validate and diagnose the process objects, and where the persistence layer can also be acceded from the Audit Layer in order to facilitate database auditing.

Auditor, Diagnoser and business processes are three parallel and "independent" systems. They are independent since they can be executed in separate machines,

Project

IdProject	hardCost	softCost	humanCost	totalCost
223	2000	2000	3000	7000
224	2000	1000	3000	6000
225	1948	2000	3000	6948
...	...	...	...	...

ProjectPerYear

IdProject	Year	IncentivePerYear	PotentialIncetive
223	2009	1000	8000
224	2010	1000	7000
225	2008	1000	7948
...	...	...	...

ProjectPerCompany

IdProject	Year	IdCompany	IncentivePerCompany
223	2009	501	5000
223	2010	745	6500
224	2009	1040	5000
224	2010	21	6800
225	2008	225	4504
...	...	...	...

Tuple A

IdProject	Year	IdCompany	hardCost	softCost	humanCost	totalCost	IncentivePerYear	PotentialIncetive	IncentivePerCompany
223	2009	501	2000	2000	3000	7000	1000	8000	5000

Tuple B

IdProject	Year	IdCompany	hardCost	softCost	humanCost	totalCost	IncentivePerYear	PotentialIncetive	IncentivePerCompany
224	2010	1040	2000	1000	3000	6000	1000	7000	5000

Tuple C

IdProject	Year	IdCompany	hardCost	softCost	humanCost	totalCost	IncentivePerYear	PotentialIncetive	IncentivePerCompany
225	2008	225	1948	2000	3000	6948	1000	7948	4504

Figure 6: Example of the obtained tuples

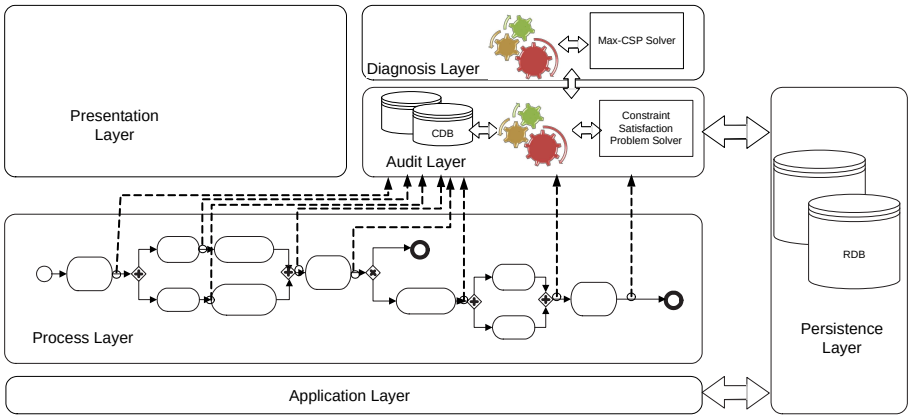

Figure 7: Framework for Run-time Auditing and Diagnoser

for different applications, and at the same time. This independence is breaking from the point of view of dataflow information, since the Auditor uses dataflow information of the process layer to detect the non-satisfiable business rules, and the Diagnosis layer uses the unsatisfiable tuples detected by the Audit layer.

The Audit layer is called from the business process layer, and depends on the business state or activity and the dataflow instances of each moment. The Diagnosis layer is call from the Audit layer, for a set of tuples and a set of business rules. In order to determine how the communication between these layers is done, it is necessary to describe some details of Audit and Diagnosis Layers.

5.1 Audit Layer

The process layer informs to audit layer about: dataflow variables and instances, the identification of the state (business execution moment) to determine which set business rules have to be analyzed, and a log file with the trace of activities that have been executed until the moment. With this information the process presented in Section 4.2 can be executed.

By using this information the business rules assume a temporal aspect (Walzer *et al.*, 2008), which means that the business rules depend on the state of the business object, hence the auditor has to be informed about which business object state to evaluate.

The CDB used in this paper is based on Labelled Object-Relational Constraint Database Architecture (LORCDB Architecture) (Gómez-López *et al.*, 2009) with an extension to represent data business object and database model relations. In our framework, the created CSP are solved with the ILOG JSolver ^{TM} tool.

5.2 Diagnosis Layer

As it has been commented, in a business process not all the activities participate in the execution of an instance. It means that not all the data are changed in the same way and not the same business rules have to be used in the evaluation of a database. While the business rules analyzed in the audit layer are only the related to the checkpoint, the business rules used by the diagnoser are all the sets of business rules that form the path for each instance in each case. The audit layer has to inform to diagnosis layer about the business rules and the unsatisfiable tuples. The Max-CSP are also solved with the ILOG JSolver ^{TM} tool.

6 CONCLUSIONS AND FUTURE WORK

In this paper the necessity to describe a methodology to audit and diagnose stored relational data in a busi-

ness process is presented. The main idea is that the business process design and implementation are tested enough, most of the problems come from to insert incorrect values in human tasks. In this paper we assume that business rules are correct and the activities work correctly, the possible incorrect things are input data. In order to describe the business rules related to stored data, Constraints have been proposed. These Constraints can be associated to different moments of a business process, in order to prevent the unnecessary evaluation of all business rules, and for all the tuples of the database.

All the mentioned ideas have been used to create a framework where an auditing and diagnosis layers have been included. The combination of CDBs together with the audit layer, enables to detect inconsistences in the relational database in function of the business rules. And the diagnoser permits to isolate the tuple and the variables that are incorrect. It is carrying out by creating and solving CSPs and Max-CSPs in execution time.

There are significant research lines that can be analyzed in further depth, such as: how would it be possible to automatically located the rules better improve the early detection of faults; and how can business rules expressed by constraints help in company decision making, proposing correct or promising values.

ACKNOWLEDGMENTS

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

REFERENCES

(Borrego *et al.*, 2009) D. Borrego, R. M. Gasca, M. T. Gómez-López, and I. Barba. Choreography analysis for diagnosing faulty activities in business-to-business collaboration. In *20th International Workshop on Principles of Diagnosis*, pages 171–178, 2009.

(Ceballos *et al.*, 2002) Rafael Ceballos, Rafael M. Gasca, Carmelo Del Valle, and Miguel Toro. Max-csp approach for software diagnosis. In *IBERAMIA*, pages 172–181, 2002.

(Chesani *et al.*, 2008) Federico Chesani, Paola Mello, Marco Montali, Fabrizio Riguzzi, Maurizio Sebastianis, and Sergio Storari. Checking compliance of execution traces to business rules. In *Business Process Management Workshops*, pages 134–145, 2008.

(Chisholm, 2003) Malcolm Chisholm. *How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

(de la Banda *et al.*, 2003) Maria Garcia de la Banda, Peter J. Stuckey, and Jeremy Wazny. Finding all minimal unsatisfiable subsets. In *PPDP '03: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming*, pages 32–43. ACM Press, 2003.

(Dechter, 2003) Rina Dechter. *Constraint Processing (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, May 2003.

(Gasca *et al.*, 2007) Rafael M. Gasca, Carmelo Del Valle, María Teresa Gómez López, and Rafael Ceballos. Nmus: Structural analysis for improving the derivation of all muses in overconstrained numeric csps. In *CAEPIA*, pages 160–169, 2007.

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

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

(Heumann, 2001) J. Heumann. Introduction to business modeling using the unified modeling language (uml). In *Rational Edge,*, 2001.

(Ma, 2007) Hongyan Ma. Process-aware information systems: Bridging people and software through process technology: Book reviews. *J. Am. Soc. Inf. Sci. Technol.*, 58(3):455–456, 2007.

(McDermid, 2003) Donald C. McDermid. Integrated business process management: Using state-based business rules to communicate between disparate stakeholders. In *Business Process Management*, pages 58–71, 2003.

(P. C. Kanellakis and Revesz, 1990) G. M. Kuper P. C. Kanellakis and P. Z. Revesz. Constraint query languages. *Symposium on Principles of Database Systems*, pages 299–313, 1990.

(van der Aalst *et al.*, 2003) Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In *Business Process Management*, pages 1–12, 2003.

(Wallace, 1995) Richard J. Wallace. Directed arc consistency preprocessing. In *Constraint Processing, Selected Papers*, pages 121–137, London, UK, 1995. Springer-Verlag.

(Walzer *et al.*, 2008) Karen Walzer, Tino Breddin, and Matthias Groch. Relative temporal constraints in the rete algorithm for complex event detection. In *DEBS '08: Proceedings of the second international conference on Distributed event-based systems*, pages 147–155, New York, NY, USA, 2008. ACM.

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

(Weske, 2007) Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.