Run-Time Auditing for Business Processes Data Using Constraints

María Teresa Gómez-López and Rafael M. Gasca

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Sevilla, Spain
{maytegomez,gasca}@us.es
http://www.lsi.us.es/~quivir

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. These data are normally stored in a relational database, and hence the database has to be analyzed to determine whether the business rules can be satisfied.

This paper presents a framework including a run-time auditing layer where the correctness of a database can be analyzed at different checkpoints of a business process according to the data flow. It provides an early detection of incorrect action on stored data. Furthermore, in order to manage the current business rules, the use of the constraint programming paradigm is proposed and the enlargement of the Constraint Database Management Systems to support business rules.

Keywords: Reasoning related to business processes, business rules, Constraint Programming.

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 very important since none of the activities of a process can work correctly using incorrect data.

For the design of a whole business process management (BPM) [1], it is necessary to design the database, the model of activities, and the causal and temporal relationships between them. Business rules can help to complete this information, since they can be used to validate business data [2]. In [3] there is a depth analysis about the integration of rule and process modelling and the shortcomings of the existing solutions. Our work is based on [4], although we propose to separate the evaluation into an independent layer that checks the business rules oriented to databases as a contract that describes the behaviour of the activities in different moments of the business process instance and only for involved data.

This paper takes a new data-oriented view of business rules engines, when there is greatest number of requirements, vast amount of data and rules. It

makes necessary to search 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 and validate business rules in function of data stored in a relational database. This paper is based on to validate *Business Data Objects*, that are defined by the set of data stored in a relational database that are updated in a business process instance. This Business Data Object is changed for the different tasks of the process, passing through different *Business Object States*. Based on these ideas, the contributions of this paper are:

- **To define a business rules language based on Constraints.** When we mention the world *Constraint* in this paper, we are talking about the constraint programming paradigm, a way to represent the correct values of a set of variables related by equations and inequations. To the best of our knowledge, no constraint satisfaction problems have been used to represent and validate business rules. Current business rules engines use the *if ... then* format to represent business rules. However by using constraints the information is represented at a more abstract level, since languages based on constraints include and improve all the capacity of representation of current rules engines, such us Drools, Fair Isaac Blaze Advisor, ILOG JRules and Jess.
- **To redefine a repository to store business rules.** If business rules are stored and well structured in a database, it will be easier to support continuous evolution of the rules in accordance with business demands, and to select which rules can be validated at any moment. To this end, we propose the use of Constraint Databases.
- **To propose a run-time auditing framework to check the conformity of the persistent data managed and data flow in a business process.** Not all the business rules are activated in the whole business process [5]. We propose a framework where it is possible to validate a set of data from a database depending on the data flow.

The combination of Constraint Databases and an audit layer permits the determination of unsatisfiable rules as soon as possible. Auditing the stored information and data flow is very important since data are normally introduced by hand. Hence, this type of population of databases produces numerous errors and inconsistent information that fluctuate. 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 is organized as follows: Section 2 gives a motivating example where constraints can be used to validate relational databases. Section 3 presents the most interesting aspects related to Business Rules and the new orientation to constraints. Section 4 explains how business rules can be stored in an efficient way. Section 5 sets out the proposed framework. Section 6 presents an extension of the constraint language for aggregate operators, and shows an example of it. Finally, conclusions and future work are presented.

2 Motivating Example of Business Process and Business Rules

The use of business rules represented as constraints has been applied to a real business process where the goal is to negotiate the collaboration projects between private companies and research groups. The persistent layer of the business process is formed of a database with 86 tables, 900 fields within these tables, more than 112.200.000 tuples, 224 triggers and 107 integrity constraints. A total of 25 employees belonging to 6 separate departments modify the stored information. In this case, the audit layer must analyze 23 states. The business process for the example is shown in Figure 1. In the given example, 270 business rules have been created where 435 variables are involved.

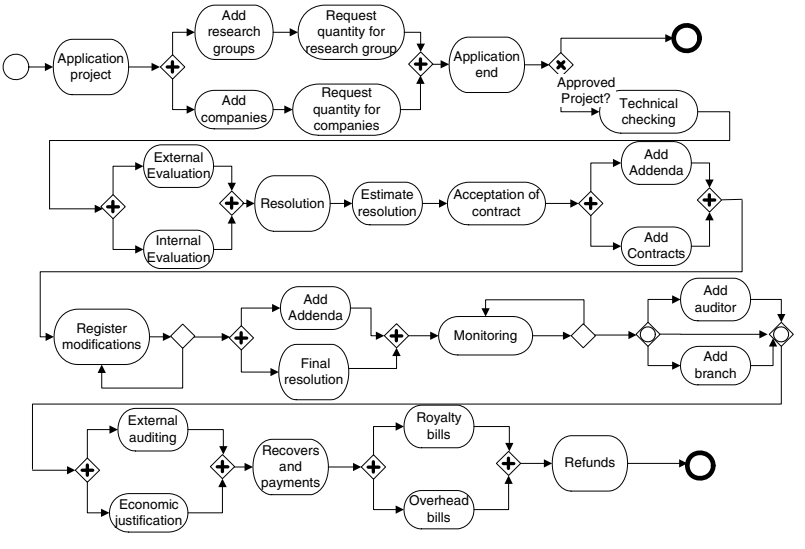

Fig. 1. Example of a real business process

This process is formed of a set of tasks which are related through various sets of business rules. Each task is in charge of a state of the project. Business rules are not described in a global way, since each task modifies certain data of a relational database and it will be necessary to evaluate different business rules.

3 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 the system. 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 [6], 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 and 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 constraints instance of the *if ... then* axiom. The 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 domains.

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 an organization policy. For example, after the *Resolution* task of Figure 1, it is necessary to check that: whether 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 or equal than 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 or equal 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 tuples with business rules that are not explicitly described. 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 $totalCost[12..161] \wedge humanCost = 1$

By using constraints and depending on the instantiation of the variables, it is possible to evaluate a tuple even if some values are not still instantiated (stored in the database), that it is equivalent to say that the value of the variable in the tuple is *null*. Hence, it permits an early detection of errors, before the whole 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 to satisfy R [7]. When an objective function f has to be optimized (maximized or minimized), then a Constraint Optimization Problem (COP) is used, which is a CSP and an objective function *f*.

By using the constraint programming paradigm, when the values of variables related to a business rule are determined in various tasks of a business process, it is not necessary to wait until all the variables are instantiated to determine whether the business rules are satisfiable. Through solving the CSP, the possible values for the variables will be found, although they are not stored in the database and cannot be inferred using only classic business rule management.

4 Database Management and Business Rules

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 data flow, 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. Metadata also describes the business data, such as definitions of business terms. In order to define the equivalence between the business rules layer and data persistence layer, the BOM (Business Object Model) was introduced [8], although not the relation between persistence layer and business layer was defined.

In order to add business rules to a business process related to its data, it is necessary to add semantic information to business rules to support database correctness.

Current architectures contain no data flow 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 data flow 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 a condition. The relation between business rules variables and database fields has to be stored. We propose the use of a system based on a wrapper over a database management system to store it and to transform business rules into query evaluations. We propose supporting the relation between business rules and persistence data through the use of Constraint Databases. These constraints can be associated to different moments of business process, in order to avoid the evaluation of all business rules, and the whole database.

4.1 Constraint Database Management System

When a great deal of business rules have to be handled, the use of a database is a mandatory decision, 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 variables and data persistence relationships. These types of information and business rules expressed by constraints can be supported by Constraint Database Management Systems (CDBMS).

Constraint Databases (CDBs) were initially developed in 1990 with a paper by Kanellakis, Kuper and Revesz [9]. The basic idea behind the CDB model is to generalize the notion of a tuple in a relational database to a conjunction of constraints, since a tuple in relational algebra can be represented as an equality constraint between an attribute of the database and a constant. In real business process, a great quantity of business rules must be defined, hence a repository is necessary in order to evaluate them as soon as possible, and to render updating easy and efficient [10].

The CDB used in this paper is based on Labelled Object-Relational Constraint Database Architecture (LORCDB Architecture) [11] with an extension to represent data business object and database model relations. LORCDB Architecture 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 2). 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. This design enables to update the business rules according to the persistence layer modifying the value in the tables of the CDBs.

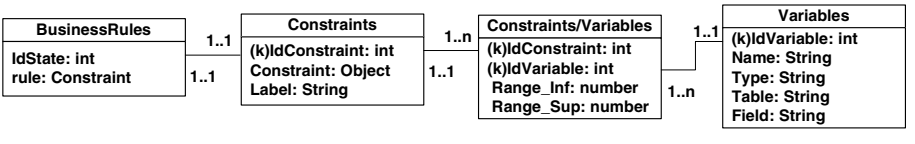

Fig. 2. CDB tables to index business rules with constraints and variables

5 Run-Time Auditing Framework

In order to permit the validation of business data in different states, and to represent and store business rules using constraints, we propose an extension of the classic Process Aware Information System (PAIS) framework [6].

Increasingly, business rules are also considered as a critical component of BPM solutions, due to the need to ensure flexibility. Some analysts believe the combination of business rules technology with BMP 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 [12], [13]. In general, a PAIS architecture

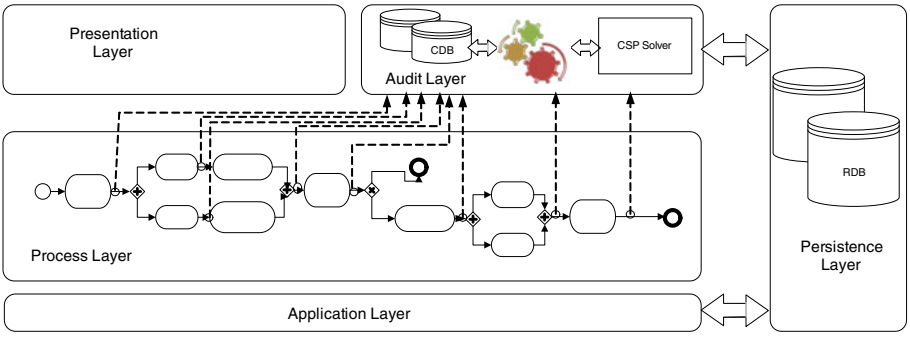

Fig. 3. Framework for Run-time Auditing

can be viewed as a 4-tier system as presented in [6], 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. We propose a new framework shown in Figure 3, where a new layer is added to validate the business data objects, and where the persistence layer can also be accessed from the Audit Layer in order to facilitate database auditing. This framework enables the relational database to be audited according to an associated set of business rules and according to the data flow.

Audit and business process layers are two parallel and "independent" systems. They are independent since they can be executed in separate machines, for different applications, and at the same time. This independence is breaking from the point of view of data flow information, since the Auditor uses data flow information of the process layer to detect the non-satisfiable business rules.

The Audit layer is called from the process layer, and depends on the business state or activity and the data flow instances of each moment. In order to determine how the communication between these layers is done, it is necessary to describe the Audit Layer in a deeper way.

Audit Layer

The function of the Audit layer is to capture the identification of the state to determine which business rules have to be analyzed, and the data flow values to delimit the tuples of the database. The behaviour of the auditor enables the determination of whether a business data object satisfies a set of rules from several points of view. The use of the audit layer implies:

- For reasons of complexity in time, it is neither possible nor necessary to analyze all the tuples of the database. In the different instances of the business process, only a set of tuples is modified. For example, if a business process is in charge of updating the information related to a project, then it is only necessary to analyze the tuples where this project is involved. This information is transferred to the Audit layer by means of the data flow. As presented in

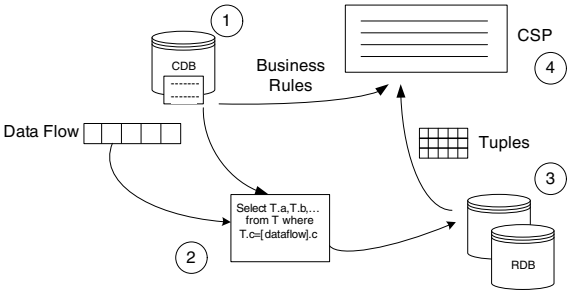

Fig. 4. Audit Layer Procedure

Section 4.1, the constraints, variables and fields of the relational database are indexed, hence a query can be created using the data flow (query condition parameters) and business rules variables.

– We propose that business rules are related to a temporal aspect [14], which means that business rules depend on the state of the business object, hence the auditor has to be informed about which business object state to evaluate. This information is used by the auditor to establish which set of rules stored in the Constraint Database has to be analyzed and combined with the tuples of the relational database to build a CSP.

Audit Layer Procedure Steps:
The steps (represented in Figure 4) to audit the data of a relational database, which depend on the data flow values and the business rules stored in the CDB, are to:

1. Select the business rules related to the state from the CDB, thereby obtaining the tables and field of relational database that are involved in the audit process.
2. Build a query where the attributes of the "projection" are the related attributes obtained in the previous step, and the condition (where) is defined by the values of the data flow.
3. Obtain the tuples that have to be evaluated, by executing the query of the previous step in the relational database.
4. Create and solve the CSPs for each tuple obtained from Step 3 and the business rules obtained from Step 1.

Applying the process to the example presented in Section 3, where the table of the database that has to be validated is presented in Table 1, the steps for the procedure become:

1. The constraints obtained to create the CSP are:
   `Float var hardCost`$[1..100]$, `softCost`$[1..150]$, `humanCost`$[1..100]$, `totalCost`$[5..250]$
   $(\texttt{hardCost} + \texttt{softCost} + \texttt{humanCost} = \texttt{totalCost} \wedge \texttt{humanCost} \leq \texttt{hardCost} * 0.10) \vee (\texttt{hardCost} + \texttt{softCost} + \texttt{humanCost} < \texttt{humanCost} \wedge \texttt{humanCost} \leq \texttt{hardCost} * 0.15)$

2. Using the table *Variables* of the CDB to know that the fields of the variables *hardCost*, $softCost$, *humanCost*, *totalCost* are respectively F_{haC}, F_{sC}, F_{huC} and F_{tC}, then the following query is created:

   `Select` $T.F_{haC}$, $T.F_{sC}$, $T.F_{huC}$, $T.F_{tC}$ `from` T `where` $T.ID$=[$dataflow$]

   and where it is supposed that the data flow has the value for identification equal to 430, and there is no condition about *year*.
3. The result of this step is the four first tuples of Table 1.
4. Each obtained tuple is evaluated by building a CSP. For the example of Table 1, the business rules are satisfiable for the first and fourth tuples. In this example, it is known that the third tuple is not satisfiable although not all its values are instantiated.

Table 1. Example of tuples to evaluate

ID	Year	*haC*	*sC*	*huC*	*tC*
430	2007	20	8	1	29
430	2008	20	7	1	15
430	2009	100	150	null	null
430	2010	100	null	4	null
431	...	...	...	...	...

6 Extending Constraint Business Rule Language with Aggregate Operators

The use of constraints to represent business rules can be extended with aggregate operators. Although other proposals exist which are oriented towards the definition of a monitoring language [15], they are not related to monitoring data flow to audit database information depending on business rules. We propose the addition of new types of business rules that can be defined over a set of tuples of relational databases: Minimum(*min(v)*), Maximum(*max(v)*), Count(*count(v)*), Summation(*sum(v)*) or Average(*avg(v)*), where v represents any variable involved in a business rule.

Going back to the example of Table 1, it is possible to define a business rule where the summation of *hardCost* for an ID has to be equal to the summation of $softCost$ for the same ID.

We have adapted the existing operators in SQL, using them to represent business rules: sum, avg, min, max and count. For each type, the creation of the model is:

- **Sum(v):** The summation will obtain the n tuples that satisfy the condition of the query. With the variables involved in the rule, and with the domain of v, the following CSP is built where i and j represent the domain of v stored in the CDB:

Integer var $\mathtt{v}_1[i..j]$, $\mathtt{v}_2[i..j]$, ..., $\mathtt{v}_n[i..j]$
$v_1 + v_2 + \ldots v_n = \ldots$

- **Min(v) and Max(v):** The minimum or maximum summation value of a variable will obtain the n tuples that satisfy the condition of the query. With the variables involved in the rule, and with the domain of v, the following COP is built where n_i represents each one of the values obtained in the tuple for variable v:

  Integer var $\mathtt{v}[i..j]$
  $v = n_1 \vee v = n_2 \vee \ldots \vee v = n_n$
  $Max(v)(orMin(v))$
  $v = \ldots$

  If some value of v is *null* for the selected tuples, the constraint $v = n_1 \vee v = n_2 \vee \ldots \vee v = n_n$ will not be included, since the non-instantiated field can take any value of the domain.
- **Count(v):** In this case, the SQL evaluation itself of the Count operator is used. The obtained value will be included in the business rules for the CSP.

Example of business rules with aggregate operators
As an illustration of our proposal, we show an example of business rules expressing by constraints for the "Acceptation of Contract" task of the business process shown in Figure 1.

```
sum(incentive) ≤ demanded
sum(incentive) = sum(potentialIncentive)∗IncentivePercentage
sum(demandPerYear) = demanded
sum(incentivePerYear) = incentive
FinalFund = FundPercentage∗demanded
max(incentive) < FundPercentage∗demanded
FundPercentage + RefundPercentage = 1
count(incentivePerYear) ≤ demanded/min(demandPerYear)
```

The CSP will be composed of any undetermined number of variables that will be established at the evaluation time, where the set of related tuples is known. Supposing that i tuples of *incentive*, j tuples of *potentialIncentive*, k tuples of $demandPerYear$ and m tuples of $incentivePerYear$ have been obtained, and the maximum value of incentive is described by the variable $incentive_i$, the minimum incentive per year $incentivePerYear_m$, and the number of years with incentive is m, then the following CSP is built:

```
Float incentive_1[domain_1], ..., incentive_i[domain_i]
Float potentialIncentive_1[domain_1]
...
Float potentialIncentive_j[domain_j]
Float demandPerYear_1[domain_1], ..., demandPerYear_k[domain_k]
Float incentivePerYear_1[domain_1]
...
Float incentivePerYear_m[domain_m], demanded[domain]
Float incentive[domain], FinalFund[domain]
Float FundPercentage[domain], ReFundPercentage[domain]
incentive_1 +...+ incentive_i ≤ demanded
incentive_1 +...+incentive_i = (potentialIncentive_1+
  ...+ potentialIncentive_j)*IncentivePercentage
demandPerYear_1 +...+demandPerYear_k = demanded
incentivePerYear_1 +...+ incentivePerYear_m = incentive
FinalFund = FundPercentage*demanded
incentive_i < FundPercentage*demand
FundPercentage + RefundPercentage = 1
m ≤ demanded/incentivePerYear_m
```

Finally, we must highlight that the CSPs cannot be pre-built and/or precompiled, since although the business rules are stored in the CDB, the number of variables and the final representation of the constraints for aggregate functions remain unknown until the time of evaluation.

7 Conclusions and Future Work

In this paper, the necessity to describe a methodology to audit stored relational data in a business process is presented. In order to describe the business rules related to the stored data, the constraint programming paradigm has been proposed. These constraints can be associated to different states of a business process, in order to prevent the unnecessary evaluation of all business rules, and the whole database.

A framework is proposed where an audit layer has been included. The combined use of CDBs and the audit layer enables early detection of incorrect data in business processes, by creating and solving CSPs and COPs in run-time.

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

Acknowledgment

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. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
2. Chesani, F., Mello, P., Montali, M., Riguzzi, F., Sebastianis, M., Storari, S.: Checking compliance of execution traces to business rules. In: Business Process Management Workshops, pp. 134–145 (2008)
3. zur Muehlen, M., Indulska, M.: Modeling languages for business processes and business rules: A representational analysis. Inf. Syst. 35(4), 379–390 (2010)
4. Meng, J.: Achieving dynamic inter-organizational workflow management by integrating business processes, e-services, events, and rules. PhD thesis, Gainesville, FL, USA, Chair-Su, Stanley Y. and Chair-Helal, Abdelsalam (2002)
5. McDermid, D.C.: Integrated Business Process Management: Using State-Based Business Rules to Communicate between Disparate Stakeholders. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 58–71. Springer, Heidelberg (2003)
6. Weber, B., Sadiq, S.W., Reichert, M.: Beyond rigidity - dynamic process lifecycle support. Computer Science - R&D 23(2), 47–65 (2009)
7. Dechter, R.: Constraint Processing. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco (2003)
8. Heumann, J.: Introduction to business modeling using the unified modeling language (uml). In: Rational Edge (2001)
9. Kuper, G.M., Kanellakis, P.C., Revesz, P.Z.: Constraint query languages. In: Symposium on Principles of Database Systems, pp. 299–313 (1990)
10. Chisholm, M.: 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 (2003)
11. López, M.T.G., Ceballos, R., Gasca, R.M., Valle, C.D.: Developing a labelled object-relational constraint database architecture for the projection operator. Data Knowl. Eng. 68(1), 146–172 (2009)
12. Ma, H.: Process-aware information systems: Bridging people and software through process technology: Book reviews. J. Am. Soc. Inf. Sci. Technol. 58(3), 455–456 (2007)
13. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York, Inc., Secaucus (2007)
14. Walzer, K., Breddin, T., Groch, M.: Relative temporal constraints in the rete algorithm for complex event detection. In: DEBS 2008: Proceedings of the Second International Conference on Distributed Event-based Systems, pp. 147–155. ACM, New York (2008)
15. Beeri, C., Eyal, A., Milo, T., Pilberg, A.: Monitoring business processes with queries. In: VLDB 2007: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 603–614. VLDB Endowment (2007)