DECISION-MAKING SUPPORT FOR THE CORRECTNESS OF INPUT DATA AT RUNTIME IN BUSINESS PROCESSES

MARÍA TERESA GÓMEZ-LÓPEZ and RAFAEL M. GASCA [*]
Departmento de Lenguajes y Sistemas Informticos, University of Seville
Seville, Spain

JOSÉ MIGUEL PÉREZ-ÁLVAREZ [†]
Intelliment Security
Tomares, Seville, Spain

In a business process, the information that flows between the activities can be introduced by those users who interact with the process. This introduced information could be incorrect due to a lack of knowledge or a mistake. For this reason and to make the business process execution consistent, we propose a Decision Support System (DSS) to inform the user about the possible and correct values that the input data can take. The DSS takes into account the business process model and the policy of the company. The policy concerning the input data and dataflow that the company manages can be represented by constraints (called Business Data Constraints). In order to ascertain all the possible values of the input data that permit the execution of the process following the defined goals, the DSS analyses the business process model and the Business Data Constraints, using the constraint programming paradigm.

Keywords: Business processes; Input Data; Decision-making support; Numerical tech-niques; Process Instance Conformity.

1. Introduction

A business process consists of a set of activities that are performed in coordination within an organizational and technical environment [1]. In a business process, the information that flows between the activities can be introduced by the users. Sometimes, the user must decide the value to introduce while taking into account the potential actions in order to make the process instance correct. In the business process scenario, this implies the analysis of all the possible branches that can be executed, and the decisions that can be taken in the future. If the decision made is incorrect, it will affect other decisions in the future, or it may even make it impos-

[*]{maytegomez, gasca}@us.es. www.lsi.us.es/∼quivir
[†]jmperez@intellimentsec.com, http://www.intellimentsec.com/

sible to finish the instance correctly (following the goals defined for the company). The requirement for decision support frequently arises when decisions have to be made in complex, uncertain, and/or dynamic environments [36]. For this reason, we propose a solution where the decision-making support for input data can be integrated into the business process instances to inform the user about the possible values of the input variables, thereby making the instance of the business process consistent. From the point of view of input data values, the correctness of a business process is based on the correctness of the compliance rules that describe the policy of the company; a correct input value is therefore the value of a variable that satisfies all the rules defined by the company.

Numerous studies propose a variety of taxonomies to classify the definition of business compliance rules [2], [3], [4], [5], 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. However if the relation between the values of the dataflow variables has to be described, then Business Data Constraints (henceforth referred to as BDCs) have the capacity to describe Business Compliance Rules [6]. These BDCs are understood as a type of business compliance rules, which represent the semantic relation between the data values that are introduced, read and modified during the execution of the business process instances [7]. The use of BDCs in decision-making support can be decisive, since humans must often make decisions about the input data in a business process instance that may result in being incorrect for the working order of the process. If all the potential scenarios in the future are taken into account in the decision-making support for input data, then late identification of non-conformities of input data with respect to the BDCs can be prevented. Therefore, we propose the use of BDCs to assist in the decisions of the users, despite the complication that the BDCs can be associated with one activity, a set of activities, or to the whole business process, and hence the assistance must integrate the model of the business process. The BDCs respond to the demand for the processing of data by providing more semantic content in the business process, which leads to a better understanding of the properties of the data used in the business process.

In order to explain our proposal, an example of web services selection is depicted in Figure 1, where a very simple business process model is shown. We have used the BPMN 2.0 notation [13] for input and output data. The BDCs are represented by means of annotations in BPMN associated to the activities. In this process, the values of data items *DE (Data Encryption)* and *NR (max Number of Requests)* are decided and introduced by a human into different activities, since they are variables that can participate in a decision-making process. The value of *TS (Throughput Saturation)* is derived by *Select Service Level*, and hence cannot be defined as an objective of the decision-making process. Depending on the value of *DE* (DE$\leq$2 or DE$>$2), one of the branches of the process is executed (XOR gateway for the different providers of services). An example of input data decision could in-

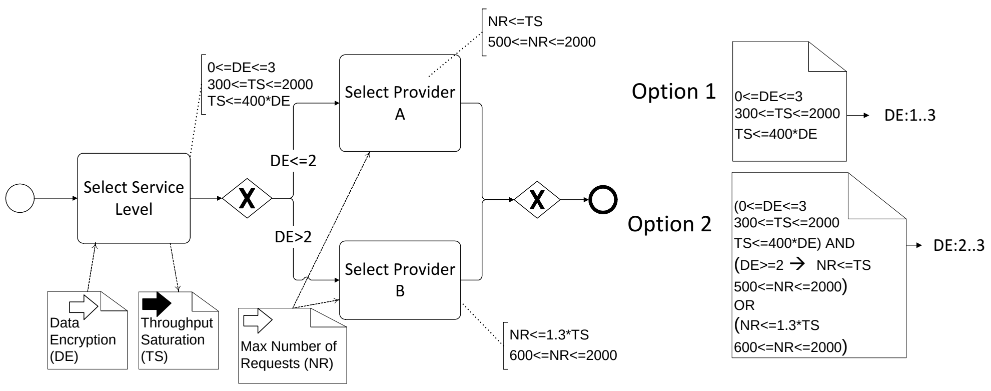


Fig. 1. Simple example of Business Process Instance for decision-making support for input data support

volve the determination of the possible correct values of *DE* in *Select Service Level* activity. As mentioned above, each BDC can be associated with one activity, a set of activities, or to the whole process: therefore each activity has several BDCs associated, for example *Select Service Level* is associated to {0≤DE≤3, 300≤TS≤2000, TS≤400*DE}. In order to decide the correct values that the variable *DE* can take to satisfy all the BDCs in the future, either those BDCs related with only *Select Service Level* (Option 1 of Figure 1) can be taken into account, or all the BDCs related directly or indirectly with *DE*. This implies analysing all the possible paths from *Select Service Level*, while considering the conditions associated with the control flows and evaluated at runtime (DE in the example), and the BDCs of *Select Provider A* and *Select Provider B* (Option 2 of Figure 1). Option 1 presents an interval with the possible values of the variable *DE* ([1..3]) when solely the BDCs related to the activity *Select Provider A* are analysed; and Option 2 shows how the interval with possible correct values of *DE* are reduced ([2..3]) if the BDCs related to the variable *NR* are also included in the analysis. For example, in Option 1 it is possible to introduce the value 1 for the variable *DE*, and to assign 350 to the variable *NR* ($\{TS \leq 400 * DE\}$). The problem arises when the activity *Select Provider A* is executed, and it is discovered that there are no valid values for the variable *NR* to satisfy the BDC {$NR \leq$ TS}.

From the previous example, it can be observed that the BDCs could help to introduce consistent data in business process instances. These BDCs constitute the basic knowledge for a Decision Support System (DSS). This knowledge can be enriched by including the analysis of the business process model, and the conditions associated with the sequence flows. This decision-making support for input data can also be used to guarantee the existence of a correct instance of the process, thereby obtaining a more fault-tolerant process.

In order to support this assistance with the DSS for input data, the following aspects have been considered:

- **The use of Business Data Constraints to help in the decision-making support for input data.** BDCs can assist in the decision-making support by reporting on the possible correct values of the data introduced in each instance; however not all of the values of the BDC variables are yet known and instantiated.
- **The development of an algorithm to traverse the process model.** To bring the relevant parts of the business process together and allow them to contribute towards decision-making support, an analysis of the process model and the BDCs of each activity is necessary. We propose an algorithm that traverses the business process model and combines the BDCs related to each activity in order to obtain a representation of the correct values that the variables can assume.
- **The generation of the Numerical representation of the possible correct values of the input data.** To generate a set of qualified solution alternatives, instead of providing only one solution, we propose obtaining the possible ranges of the decision variables by means of Constraint Programming.
- **The implementation of an application to integrate our proposal into business process modeller software.** In order to facilitate the decision-making process description, we have developed an application for the DSS (called MARTIN: MAking Reasoning for daTa INput) that allows the user to define the BDCs for each activity, and connect the symbolic and numerical solvers with the process in order to obtain the possible correct values for the input data in each instance. This application offers the required agility and flexibility to the organizations so that they can comply with changes in policy and legislation, and apply them in the DSS.

For these reasons, this paper is organized as follows: Section 2 explains the grammar of BDC used in this paper. Section 3 presents a motivating example where decision-making support for input data is used. Section 4 states the necessary definitions to formalize the proposal. Section 5 analyses how to traverse a business process model to study all the possible correct values of a variable, and to propose an algorithm for its development. Once the BDCs involved in the decision-making support are known, how Constraint Programming is applied to Decision-Making support is analysed in Section 6. In Section 7, the details of the DSS that we propose, the implemented application to include BDCs in the decision-making support, and a case of study are explained. Section 8 discusses previous work related to our proposal. Finally, conclusions are drawn and future work is presented.

2. Representation of the Decision-Making Objectives

As mentioned above, BDCs are a subset of the business compliance rules, oriented to dataflow values. BDCs can be used to represent the relation between the values of the variables that flow in a business process instance, by helping in the description of the policy of the company and in the decision-making support for input data. Although there are several language constructs for the design of business compliance rules, most are based on the use of IF-THEN rules or their derived extensions of ECA rules (event-condition-action rules) or ECAA rules (event-condition-action-alternative rules). The languages for representing business rules vary between research prototypes (e.g. N3), vendor specific formats (e.g. Drools, Fair Isaac Blaze Advisor, ILOG JRules and Jess), and proposals for the XML-based exchange of business rules (e.g. SRML, PRR, and SBVR). Another possibility is the Business Process Compliance Language (BPCL) [8], 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 [6] to develop a semantic approach to business rule management that allows intuitive modelling and analysis of business process compliance. In [7], a grammar for BDCs is presented. In this paper, we extend this grammar to include new operators that can facilitate the description of the policy of the companies. This extension is inspired by the idea of viewing a BDC as a *Constraint*: a Boolean combination of equations and inequations that follows the metamodel of Figure 2 based on that presented in [9].

A BDC can be an atomic constraint, a negation of a constraint, or a binary constraint formed by two constraints joined with a Boolean operator (AND, OR, IMPLY). An atomic constraint is formed by two functions and a comparator ($<$, $\leq$, $\geq$, $\ldots$). Each function can be a unary function (a variable), a constant, or a binary function joined with an operator ($+$, $*$, $-$, $/$).

The metamodel is based on the recursive definition of Constraint, since two constraints can be combined into a new one, by using one of the Logic Operators or a Negation. For example, the constraints: $\{a+b \leq c \wedge c > 8\}$ and $\{a*b \geq 5 \vee \neg(c \leq 15)\}$, can be combined into a new constraint, for example by using the $\vee$ operator, thereby obtaining the new constraint: $\{(a+b \leq c \wedge c > 8) \vee (a*b \geq 5 \vee \neg(c \leq 15))\}$. We therefore define four methods for the combination of a BDC (c) with another (c'):

- c.and(BDC c') returns a new constraint: $\{c \wedge c'\}$
- c.or(BDC c') returns a new constraint: $\{c \vee c'\}$
- c.not() returns a new constraint: $\{\neg\, c\}$
- c.imply(BDC c') a new constraint: $\{c \rightarrow c'\}$

These methods will be used in Algorithm 5.2 to build the BDC that will represent all possible values of a variable in accordance with the model of the business process.

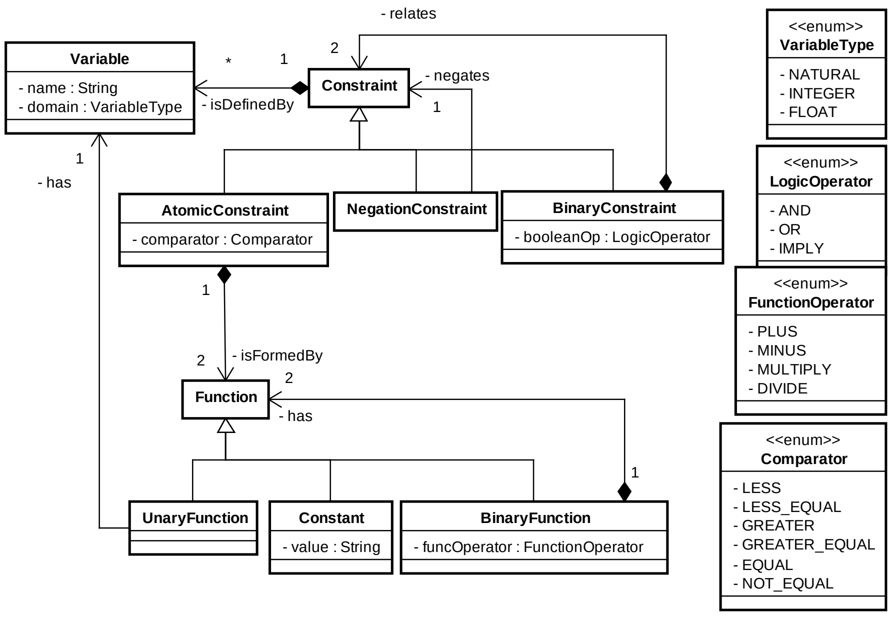


Fig. 2. Constraint Metamodel

3. A motivating example

The motivating example used in this paper is the well-known example of the organization of a conference, for which a reduced model is presented in Figure 3. This business process shows an example where decisions about input data must be made at various points of the business process, where the future values of several variables remain unknown, since they are introduced into activities that have yet to be executed. First of all, the organizing committee has to determine the early and late registration fees several months before the number of participants is known, and this decision cannot be changed after the call for papers has been made. A similar situation exists when deciding the number of proceedings that will be printed, but although the final number of participants remains unknown until the conference ends, there is a significant relationship between the number of accepted papers and the number of participants. Other decisions, such as which restaurant to book for the gala dinner and lunches, have to be made although the number of participants can influence the determination of the price and the choice of restaurant. Although this information is unknown, it is necessary to make many decisions before the conference starts. In the example, nine different input data variables participate in the data decision-making process (early registration fee, late registration fee, lunch price, dinner price, number of proceedings to print, local speaker cost, international speaker cost, publicity and venue cost). There are other input variables, but their

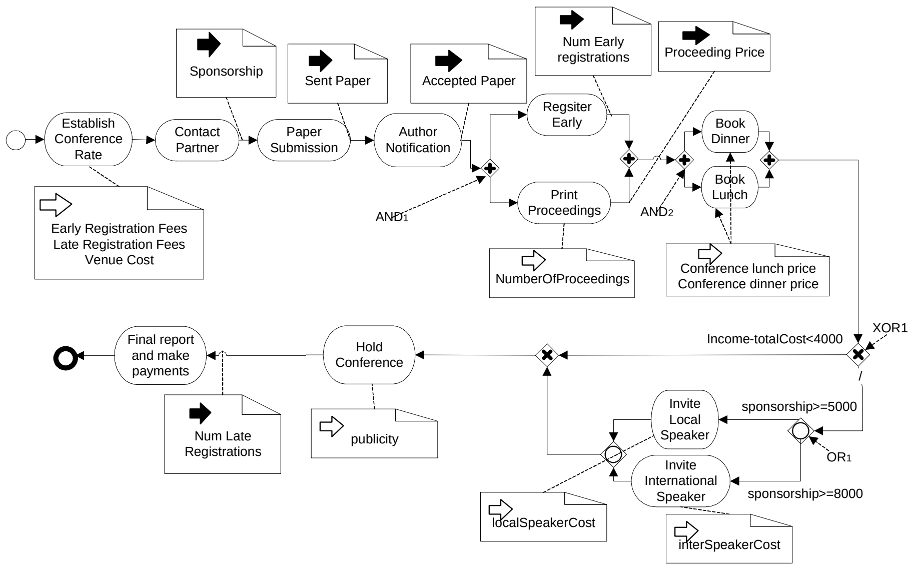


Fig. 3. Example of a conference organization process

values are determined in a mandatory way, for example the number of accepted papers, or the final number of participants that belong as part of the data flow. Since decisions have to be made by the organizing committee, two questions arise: Would it be more conducive towards the success of the conference to take into account all the potential facts in the future, and the possible branches that will be executed depending on the specific values of each process instantiation? How it can be done?

Obviously, if there is no information about the relationship between the variables *earlyRegistrationFee*, *lateRegistrationFee*, *lunchPrice*, etc, no type of inference about the possible correct values can be taken. For this reason, BDCs are necessary. Examples include:

- $\{$sentPaper*0.3 $\leq$ acceptedPapers $\leq$ sentPaper*0.8$\}$ associated with the activity **Paper Submission**.
- $\{$numEarlyParticipant*0.8 $\leq$ acceptedPapers $\leq$ numEarlyParticipant * 1.2$\}$ associated with the activity **Author Notification**.
- $\{$numOfProceedings $\geq$ 1.1 * numberOfParticipants$\}$ associated with the activity **Print Proceedings**.
- $\{$Income*0.80 $\leq$ TotalCost $\leq$ Income*0.9$\}$ associated with the activity **Hold Conference**.

These BDCs, combined with the process model and the conditional sequence flows presented in Figure 3, can be applied in order to ascertain the decision variables, for example *lunchPrice* in the activity **Book Lunch**. All the variables and

BDCs are explained in detail in the Appendix of the paper.

4. Formalization of the Decision-making support for input data in Business Processes

In this section, basic notation and concepts are introduced and the formalism used to express our problem is briefly described. A DSS for input data in business processes undertakes tasks such as an evaluation of alternatives, and communicates its conclusions by taking into account the BDCs that must be consistent in future decisions and in the process model. Therefore, the same model can be involved at different points of decision-making according to the decision variables that the user needs to ascertain, and to the specific values instantiated for the variables of the dataflow. Therefore, two aspects are combined in the decision-making support: the model $\mathbb{M}$ (<$\mathbb{P}$, $\mathbb{BDC}$, $\mathbb{DF}$>), formed by activities and control flows, the BDCs associated to each activity, and the dataflow variables; and the decision point $\mathbb{DP}$, formed by the decision variables (whose possible values need to be ascertained) and all the instances of the dataflow variables until the moment <$\mathbb{DV}$, $\mathbb{DFI}$>. According to these two descriptors, the decision-making support can be performed by obtaining all the possible $\mathbb{DVI}$ tuples of values for $\mathbb{DV}$ that satisfy $\mathbb{DBC}$ and the $\mathbb{DFI}$. A DSS for the input data of a business process can therefore be specified by means of the Decision Process Model:

Definition of Decision Process Model: This is formed by means of the business process model ($\mathbb{M}$), and the decision points ($\mathbb{DP}$). Each of these parts are defined at the same time as:

$\mathbb{M}$ = <$\mathbb{P}$, $\mathbb{BDC}$, $\mathbb{DF}$>

$\mathbb{DP}$ = <$\mathbb{DV}$, $\mathbb{DFI}$>

By using these two parts of the description, the decision-making support for input data obtains the $\mathbb{DVI}$, which is a set of correct tuples of instantiation of $\mathbb{DV}$:

$<\mathbb{M}, \mathbb{DP}> \rightarrow \mathbb{DVI} \mid \forall \ \mathrm{vd}_i \in \mathbb{DVI}, \{\mathbb{BDC} \cup \mathbb{DFI} \cup \mathrm{vd}_i\} \vdash \top$

The following subsections formalize the model, the decision points, and specify how to obtain the decision-making support in detail.

4.1. *Business Process Model ($\mathbb{M}$)*

A part of the Decision Process Model is the process model $\mathbb{P}$, and is composed of:

- $\mathbb{SE}$, one start event to initialize the process.
- $\mathbb{EE}$, a set of end events, with at least one element.
- $\mathbb{A}$, a set of activities that defines the model of the process.
- $\mathbb{CF}$, a set of control flow patterns (AND, OR, XOR) that describes the possible branches to execute.
- $\mathbb{C}ond$, a set of conditions associated with the control flows OR and XOR, that describes the paths that the process can take depending on the values of the variables in the dataflow. These conditions are evaluated at runtime,

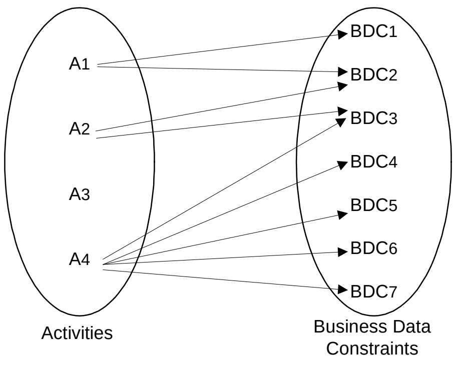


Fig. 4. Function relation between Activities and BDCs

when the values of the variables are known.

The BDCs associated to each activity of the process ($\mathbb{BDC}$) are defined with the dataflow variables $\mathbb{DF}$; the $\mathbb{DF}$ can then be defined for a range of possible values to assist in the decision support ($\mathbb{DFR}$). The relationship between activities and BDCs is presented in Figure 4, where there is a surjective correspondence between the Activity and the Business Data Constraint sets. The importance of the integration of business rules and business processes was analysed in [33]. Every BDC$_i$ in a Business Data Constraint set has a corresponding element A$_j$ in the Activity set, such that f(A$_j$) = BDC$_i$, whereby multiple activities might be turned into the same BDC by applying f, although not all the Activity elements must have a relation with the elements of the Business Data Constraint set. Therefore, the $\mathbb{BDC}$ is described by means of the tuple: $\mathbb{BDC} = < \mathbb{DFR}, f : A \rightarrow BDCs>$.

The BDCs that describe the range of the possible values for each dataflow variable are assigned to the whole process.

For the process example presented in Figure 3, the components of the process according the definitions above are:

- $\mathbb{A}$ = {Establish Conference Rate, Contact Contribution, Paper submission, Author Notification, ...}.
- $\mathbb{CF}$ = {AND$_1$, AND$_2$, XOR$_1$, OR$_1$}.
- $\mathbb{C}ond$ of the control flow pattern XOR$_1$ is {Income-totalCost $<$ 4000}, and for the control flow pattern OR$_1$ are {sponsorship $\geq$ 5000, sponsorship $\geq$ 8000}.
- $\mathbb{BDC}$ is composed of:
  - $\mathbb{DFR}$ = {5000 $\leq$ totalCost $\leq$ 50000}, {5000 $\leq$ income $\leq$ 60000}, {50 $\leq$ numEarlyParticipants $\leq$ 200}, ...
  - $f : A \rightarrow BDCs$ = **Establish Conference Rate**$\rightarrow$ {1.2 * earlyRegistrationFee $\leq$ lateRegistrationFees $\leq$ 1.5 * earlyRegistrationFee, fix-

Cost = venueCost + publicity},
Paper Submission$\rightarrow$ {sentPaper*0.3 $\leq$ acceptedPapers $\leq$ sentPaper*0.8}, ...

- $\mathbb{DF}$ = {earlyRegistrationFee, lateRegistrationFee, sponsorship, acceptedPaper, numEarlyParticipants, numberOfProceedings, proceedingsPrice, ...}

4.2. *Decision Points for Decision-Making Support*

For the same process model and BDCs, various evaluations can be obtained, depending on the decision variables for which the user requests assistance, and the instantiations of the dataflow that are defined. Therefore, the decision variables $\mathbb{DV}$ constitute a set of input data (vd$_1$, vd$_2$, ..., vd$_n$), and the $\mathbb{DFI}$ is a partial assignment of values to $\mathbb{DF}$.

For the example of Figure 3, one decision point could be:

- $\mathbb{DV}$ = {numberOfProceedings}
- $\mathbb{DFI}$ = {500, 600, 2500, 123, 56, 60, *null*, *null*, ...}, which is a partial assignment of the set $\mathbb{DF}$ = {earlyRegistrationFee, lateRegistrationFee, sponsorship, acceptedPaper, numEarlyRegistration, numberOfProceedings, proceedingsPrice, ...} carried out during the partial execution of the business process up to this decision point.

4.3. *Support obtained for the Decision-Making for Input Data*

In a consistency-based approach, a decision-making support for input data returns, for the decision variables $\mathbb{DV}$ = {dv$_1$, dv$_2$, ..., dv$_n$}, a set of m tuples (one set for each possible tuple of values), where each tuple has n values, which are as many values as they are decision variables $\mathbb{DV}$. Therefore, the tuples returned have the form: $\mathbb{DVI}$ = {<dv$_1^1$, dv$_2^1$, ..., dv$_n^1$>, ..., <dv$_1^m$, dv$_2^m$, ..., dv$_n^m$>}, where dv$_i^j$ represents a possible value of instantiation of the decision variable dv$_i$ in the j-th tuple of the possibilities presented.

The obtained $\mathbb{DVI} \subseteq$ DV$_1$ $\times$ ... $\times$ DV$_m$, where DV$_i$ represents the domain of dv$_i$, and $\forall$ $dv \in \mathbb{DVI}$, $\{dv \cup \mathbb{BDC} \cup \mathbb{DFI}\} \vdash \top$.

The problem is that showing the user all these possible $\mathbb{DVI}$ tuples, which represent the possible correct values of the decision variables, would not be very helpful, since the combination of possibilities can be very large, or even infinite, and difficult to analyse. For this reason, we propose presenting the information by means of intervals for each decision variable:

Since the set of tuples $\mathbb{DVI}$ is equal to {<dv$_1^1$, dv$_2^1$, ..., dv$_n^1$>, ..., <dv$_1^m$, dv$_2^m$, ..., dv$_n^m$>}, then the interval representation returns a set of intervals of the form:

<[dv$_1^e$ .. dv$_1^i$] .. [dv$_1^h$ .. dv$_1^j$], [dv$_2^e$ .. dv$_2^i$]..[dv$_2^h$ .. dv$_2^j$], ..., [dv$_n^e$ .. dv$_n^i$]..[dv$_n^h$ .. dv$_n^j$]>, where $\exists\{v_1, v_2, ..., v_n\}$ | $v_1 \in$ [dv$_1^e$ .. dv$_1^i$] $\cup ... \cup$ [dv$_1^h$ .. dv$_1^j$], $v_2 \in$ [dv$_2^e$ .. dv$_2^i$] $\cup ... \cup$ [dv$_2^h$ .. dv$_2^j$], ..., $v_n \in$ [dv$_n^e$ .. dv$_n^i$] $\cup ... \cup$ [dv$_n^h$ .. dv$_n^j$].

The tuple of values $\{v_1, v_2, ..., v_n\}$ satisfies all the BDCs, thereby assuring that there is a tuple of values contained in the intervals for which the instance can be executed correctly according to the BDCs. It should be noted that the BDCs are not always satisfies by all the possible combinations of values for $\{v_1, v_2, ..., v_n\}$. To assure that any combination of input values for decision variables is consistent at a decision point, only one decision variable can participate at the same time, since it would be possible to violate a BDC if more than one value were introduced.

To summarize, the decision-making support for input data is developed in two phases:

(1) **Obtaining the process model at design time for the decision-making support**. Although the parts of the model have been described, the way in which the combination of the BDCs related to the activities is achieved needs to be explained. Section 5 presents how to obtain a new BDC that represents all potential decisions. This algorithm takes into account the business process model, the conditions associated with the control flows, and the BDCs related to each activity. This step has to be reexecuted whenever the business process model or the BDCs change.

(2) **Evaluating the decision points at runtime for the decision-making support.** The output of the DSS will depend on the decision point, that is, on the decision variables and the instantiation of the dataflow variables. How to obtain the numerical representation is explained in 6.

4.4. *Scope of Applicability of our Proposal*

As a consequence of the Business Process Model described in the previous subsections, the scope of application of our proposal is limited by three characteristics:

- **The knowledge concerning the possible values of the variables managed during the business process instance**. If the possible values of the variables, and their relations, remain unknown, it will not be possible to help the user during the business execution. The inference of the possibility of correct input values is derived from the Business Data Constraints associated to the model.
- **The possible BDCs that can be used, since the limitation is centred on the capacity of their expressiveness as allowed by the grammar and type of variables**. These BDCs constitute the formal representation of the relations between the data that forms the business process. The limitations of use of the proposal appear when the constraints cannot be represented by numerical relations, data type, or the operators included in this proposal, such as when a relation among 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 used for the Constraint Programming Problem, explained in Section 6. Most of the commercial solvers maintain the capacity

to include Float, Integer, Sets, and Boolean variables in the model, thereby making it possible to cover a significant number of problems and their business data constraints.

- **The model of the process ($\mathbb{M}$) follows the BPMN representation, but only with a subset of artefact**. The components allowed are: activities, control-flows (AND, OR and XOR, split and join), sequence flow, flow-condition, one stat event, and end events. Only the process models that can be transformed into a correct process graph (as defined in Section 5) can be studied.

5. Obtaining the BDCs from the Decision Process Model

As stated earlier, since the BDCs tend to be associated with various activities in a business process, then the BDCs associated with the activity where the decision point is executed are not the only influence on the possible values of a data introduced in this activity. The question therefore becomes: in which sense can the business process model and the conditions associated with the control flows influence the decision? By analysing the most common patterns of control flow in business process models (Sequence, AND, OR, and XOR), the possibilities are determined as:

- **Sequence.** By using the example of Figure 6, in order to ascertain the possible values of the variable *earlyRegistrationFee* in the Activity *Establish Conference Rate*, then the BDCs of the activities *Contract Partner*, *Paper submission*, ... have to be taken into account with an AND Boolean relation between their BDCs, since all the instances must execute these activities, and satisfy their BDCs (Figure 5.a).
- **AND Split.** Similarly with the AND split control flow, all the instances have to execute all the activities of the different branches, although the order is unknown. Therefore, the BDCs of all these activities will be combined by means of an AND Boolean combination, as presented in Figure 5.b.
- **XOR Split.** In the case of the XOR control flow, where only one branch can be executed for an instance, the condition associated with each branch will be combined with the BDCs of the activities for each branch. The BDCs of the activities of a branch have to be satisfied only if the branch is executed: this occurs when the condition associated to the XOR control flow is *true* for the values of the instantiation. For the example of Figure 6, the Activities *Local invitation* or *International invitation* can only be executed if the condition $\{\text{sponsorship} \geq 8000\}$ is true, and hence the correctness of the BDCs of these activities is conditioned to the value of the variable *sponsorship*. As presented in Figure 5.c, the BDCs of the activities of the different branches will have an OR Boolean relation between them, and the conditions are combined with an AND Boolean relation with the BDCs of the activities for each branch. A special treatment is performed for the *default* branch, where the conditional

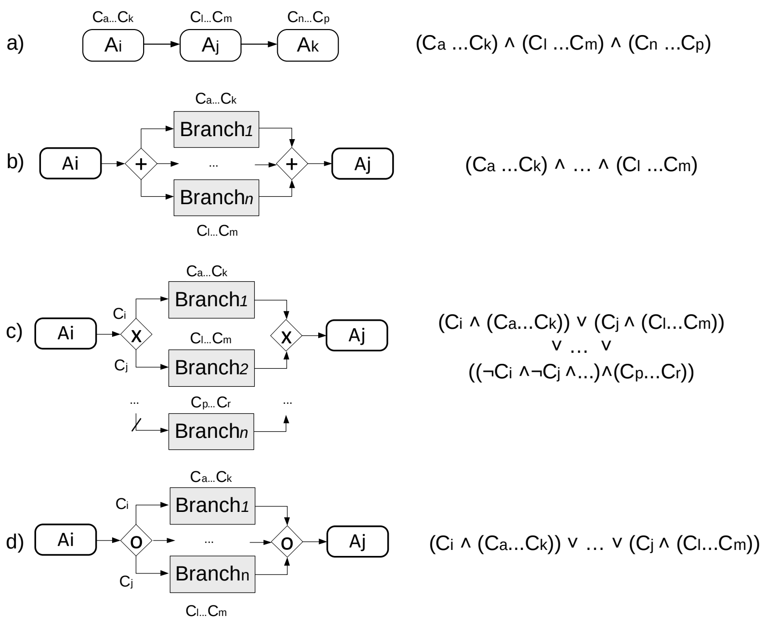

a)
Ca...Ck
Cl...Cm
Cn...Cp
Ai
Aj
Ak
(Ca ...Ck) ∧ (Cl ...Cm) ∧ (Cn ...Cp)
b)
Ai
Branch1
Branchn
Aj
Ca...Ck
Cl...Cm
(Ca ...Ck) ∧ ... ∧ (Cl ...Cm)
c)
Ai
Ci
Cj
Branch1
Branch2
Branchn
Aj
Ca...Ck
Cl...Cm
Cp...Cr
(Ci ∧ (Ca...Ck)) ∨ (Cj ∧ (Cl...Cm))
∨ ... ∨
((¬Ci ∧¬Cj ∧...)∧(Cp...Cr))
d)
Ai
Ci
Cj
Branch1
Branchn
Aj
Ca...Ck
Cl...Cm
(Ci ∧ (Ca...Ck)) ∨ ... ∨ (Cj ∧ (Cl...Cm))

Fig. 5. Combination of BDCs in terms of the control flows and their conditions

flow of execution implies the non-compliance of the condition for the rest of the branches, thereby implying the negation of the rest of the conditions.

- **OR Split.** OR control flow is very similar to XOR, the only difference being that more than one branch can be executed, and hence the *default* option negating the rest of the branches does not appear since this would make no sense, as presented in Figure 5.d.

In order to traverse the model, by combining the structures with the concepts of the control flows presented above, we propose building a process graph that is used to represent the business process details. The construction of the process graph is based on the annotated graph presented in [10], but includes some differences: for example it employs a different way to label the graph to model the conditions associated with the branches for the OR and XOR splits, and varies the number of possible end nodes.

Definition of a Process Graph: A process graph is a labelled directed graph composed of nodes (N) and edges (E), $G = <N, E>$, where N is the disjoint union of $\{n_0\}$ (start node), N_+ (end nodes), N_T (task nodes), N_{PS} (parallel splits), N_{PJ} (parallel joins), N_{ORS} (or splits), N_{ORJ} (or joins), N_{XS} (xor splits), and N_{XJ} (xor joins). For $n \in N$, IN(n)/OUT(n) denotes the set of incoming/outgoing edges of n.

In order to determine that a business process model described by a process graph is correct, it is required that:

(1) For each split node n, $|\text{IN}(n)| = 1$ and $|\text{OUT(n)}| > 1$;
(2) For each join node n, $|\text{IN}(n)| > 1$ and $|\text{OUT}(n)| = 1$;
(3) For each $n \in N_T$, $|\text{IN}(n)| = 1$ and $|\text{OUT}(n)| = 1$;
(4) For n_0, $|\text{IN}(n)| = 0$ and $|\text{OUT}(n)| = 1$, and vice versa for $n \in N_+$;
(5) Each node $n \in N$ is on a path from the start to an end node.
(6) If $|\text{IN}(n)| = 1$, then $\text{IN}(n)$ is identified with its single element, and similarly for $\text{OUT}(n)$;
(7) The outgoing edges of $n \in \{N_{XS} \cup N_{ORS}\}$ have to be labelled with a condition to describe when this branch is executed.
(8) One and only one of the labels for the outgoing edges of a node $\in N_{XS}$ can be labelled as *default*, or none of them can be labelled as *default*.

We suppose that the business process model is correct, which implies that it follows the aforementioned rules and that it demands that the conditions of the XOR branches cannot be overlapped, and that all the splits are closed in a join or in an end node. The graph that we obtain from the example of Figure 3 is presented in Figure 6.

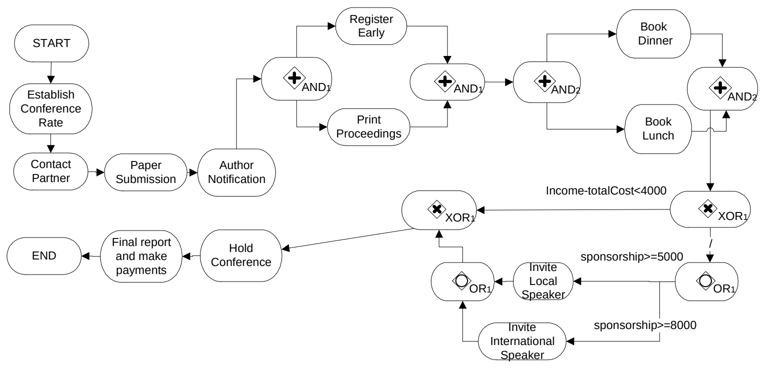


Fig. 6. Process Graph for the Conference example

Once the graph is created, the next step is to ascertain how to traverse the graph to combine all the BDCs of its activities, to obtain the model that represents the possible values of the dataflow variables, by including the topological characteristics explained in this subsection. To this end, we propose an algorithm to traverse a process graph and to obtain a BDC that represents, in a symbolic way, all the valid values of the decision variables. We have developed the recursive Algorithm GraphTraversal (Algorithm 5.2), whose complexity is linear for the number of nodes, since each node is analysed only once. The input parameters of Algorithm 5.2 specify the graph to traverse, the node used to start the algorithm, and the input/output constraint that represents the BDC obtained from the execution of the algorithm. The output of the recursive function consists of the subsequent node analysed in the process of traversing. In order to initialise the call to recursive Algorithm 5.2, Algorithm 5.1 has been defined, which starts with the necessary first call.

The line of reasoning in the algorithm follows the definition of the process graph presented in the previous section, where the graph has special characteristics, for example there is no node n with $|\mathrm{IN}(n)| > 1$ and $|\mathrm{OUT}(n)| > 1$, and for all $n \in N_T$, $|\mathrm{IN}(n)| = 1$ and $|\mathrm{OUT}(n)| = 1$. This algorithm enables us to approach a business process as a sequence of activities, where each activity itself can also be a process graph (a subprocess). In order to traverse the graph, the list of activities or subprocesses that form it must be looped in (line 2 of Algorithm 5.2). The problem is how to detect the subprocesses: this implies determining where they start, and where they end. We define that a subprocess starts when a split control flow is found in the traversing process (line 4 of Algorithm 5.2), and ends when a join control flow (line 33) or an end node (line 2) is found. When a subprocess is detected, a recursive call is used to traverse each branch (line 20). The outputs of these recursive calls are also BDCs, which are combined according to the control flow and the conditions (lines 18 and 24), following the instructions of Figure 5 (from line 20 to 31). Since it is supposed that the graph represents a correct model, all the branches have to finish in the same join control flow, or in an end node. For this reason, when all the recursive calls for the different branches end, and by using all the nodes returned from these calls, the node which is distinct from an end node will be used as the next node in the loop (line 32).

Algorithm 5.1 Algorithm to initialize the traversing of a process graph

```
1: function GraphTraversal(Graph g)
2:     Constraint c = new Constraint();
3:     Node n = g.obtainNeighbour("Start");
4:     GraphTraverse(g, n, c);
5:     return $c$
6: end function
```

The BDC obtained for the graph of Figure 6 is the following, where, for the sake of simplicity, the BDCs of the activities are presented as the name of the activity:

Establish Conference Rate ∧ *Contact Partner* ∧ *Paper Submission* ∧

Author Notification ∧ (*Early Registration* ∧ *Print Proceedings*) ∧

(*Book Dinner* ∧ *Book Lunch*) ∧

(((Income-totalCost<4000) ∧ (true)) ∨ (¬(Income-totalCost<4000) ∧ ((sponsorship≥5000)∧(Local invitation) ∨ ((sponsorship≥8000) ∧ (International invitation))))) ∧

Hold Conference ∧

Algorithm 5.2 Recursive Algorithm to t a process graph

```
function GraphTraversal(Graph g, Node n, Constraint c)
    while n is not an END node do
        if n is an Activity then
            if c == new Constraint() then
                c = n.obtainConstraint();
            else
                c.and(n.obtainConstraint());
            end if
            n = g.obtainNeighbour(n);
                    ▷ A single node is returned (for line 3 of Process Graph definition).
        else if n.type is a Split Control Flow then
            Set nodes = g.obtainNeighbour(n);
                                ▷ Several neighbours are obtained, one for each branch.
            Node n1 = nodes.get();
            Constraint c1 = new Constraint();
            Array ArrayNodes[] = new Nodes[nodes.size()];
            ArrayNodes[0] = GraphTraversal(g, n1, c1);
            c1.add(g.label(n,n1));
            integer i = 1;
            while nodes.next() do
                Node n2 = nodes.get();
                Constraint c2 = new Constraint();
                ArrayNodes[i++] = GraphTraversal(g, n2, c2);
                c2.and(g.label(n, n2));
                if n is an OR or an XOR control flow then
                    c1.OR(c2);
                else
                    c1.AND(c2)
                end if
            end while
            c.and(c1);
            n = theNodeDistinctOfEnd(ArrayNodes);
        else                                                        ▷ Any join control flow
            n = g.obtainNeighbours(n);
                    ▷ A single node is returned (for line 2 of Process Graph definition).
            return n;
        end if
    end while
    return n
end function
```

Final report and make payments

Any BDCs defined for the whole process, and not just for a specific activity, will be included with an AND Boolean relation with the BDCs obtained from the execution of the algorithm. If there is an activity without any associated BDC, it will be equal to not including a constraint or including a *true* constraint in the BDC obtained.

6. Evaluating the Decision Process Model at the Decision Points

As mentioned in Section 4, an important key to the decision-making support for input data is how to present the information so that it is useful to the user. In order to obtain a numerical representation by means of intervals, we propose the use of the Constraint Programming paradigm to assure that the decision model can be evaluated in an efficient way. This assurance is thanks to the CSP formal representation being very similar to the formal representation of the decision model presented in this paper. Therefore, we propose modelling and evaluating a Constraint Satisfaction Problem with the BDC obtained from Algorithm 5.2, and for the instantiated variables of dataflow.

A Constraint Satisfaction Problem (CSP) represents a reasoning framework consisting of variables, domains and constraints. Formally, it is defined as a tuple $<X, D, C>$, 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. Each constraint C_i is defined as a relation R on a subset of variables $V = \{x_i, x_j, \ldots, x_l\}$, called the *constraint scope*. The relation R may be represented as a subset of the Cartesian product $d(x_i) \times d(x_j) \times \ldots \times d(x_l)$. A constraint $C_i = (V_i, R_i)$ simultaneously specifies the possible values of the variables in V in order to satisfy R. Let $V_k = \{x_{k_1}, x_{k_2}, \ldots, x_{k_l}\}$ be a subset of X, and an l-tuple $(x_{k_1}, x_{k_2}, \ldots, x_{k_l})$ from $d(x_{k_1}), d(x_{k_2}), \ldots, d(x_{k_l})$ can therefore be called an *instantiation* of the variables in V_k. An instantiation is a solution if and only if it satisfies the constraints C.

In order to solve a CSP, a combination of search and consistency techniques is commonly used [11]. The consistency techniques remove inconsistent values from the domains of the variables during or before the search. During the search, a propagation process is executed which analyses the combination of values of variables where the constraints are satisfiable. Several local consistency and optimization techniques have been proposed as ways of improving the efficiency of search algorithms.

In a CSP, the inclusion of a constraint in the set C has the same effect as including this constraint with an $\wedge$ relation with the set C. For this reason, in this case the CSP will be composed of the variables of the dataflow, both instantiated and non-instantiated, of the BDC obtained from Algorithm 5.2, and of the BDCs defined for the whole process. The parts of the CSP according to the definition of Decision Process Model are therefore:

- X: $\mathbb{DF}$
- D: $\mathbb{DFI}$
- C: {BDCs defined for the whole process} $\cup$ {BDC obtained from the execution of the algorithm GraphTraversal}

Since the CSP returns all the possible values of the variables ($\mathbb{DF}$ in this case), it is necessary to reduce it to present only the values of the decision variables ($\mathbb{DV}$). To this end, the decision variables are defined as objectives during the propagation process where the variables are instantiated. This enables the search to stop the instantiation in the branches where no new values of decision variables can be found, thereby bounding the unnecessary combinations of values. For each solution found, each value of the decision variables is stored in a sorted list. Each of these sorted lists is treated in order to return the list of intervals for each variable of decision. For example if the values {1, 2, 3, 5, 8, 9, 10} are found for the variable *x*, the list of intervals built is {[1, 3], [5, 5], [8, 10]}.

For the example of Figure 3, the CSP built to analyse the possible valid values of the variable of decision *NumberOfProceedings* in the activity *PrintProceeding* (which uses the variables and BDCs presented in the Appendix) is:

```
//All the variables of the dataflow
totalCost, numEarlyParticipant, numLateParticipant, ... Integer
//The variables instantiated until the decision point
EarlyRegistrationFee = 500
LateRegistrationFee = 750
...
//Range of the dataflow variables and BDCs for the whole process
totalCost[5000..50000] Integer
numEarlyParticipants[50..200] Integer
numLateParticipants[10..100] Integer
numParticipants[60..300] Integer
CostPerParticipant = 3*lunchPrice+dinnerPrice+proceedingsPrice
...
//BDCs obtained from the algorithm GraphTraversal
BDCs of the Activity Establish Conference Rate
BDCs of the Activity Contact Partner
...
(((Income-totalCost<4000) ∧ (true)) ∨ (¬(Income-totalCost<4000) ∧
(sponsorship≥5000)∧(Local invitation) ∨
   ((sponsorship≥8000) ∧ (International invitation))))
//BDCs of the Activity Hold Conference
//BDCs of the Activity Final report and make payments
Goal for branching(numOfProceedings)
```

The CSP solver used in our proposal is Choco^{TM} [12]. Once the resolution of the CSP has finished, the list of intervals obtained is used to inform the user about the possible correct values. For the example, once *earlyRegistrationFees*, *lateRegistrationFees*, *venueCost*, *sponsorship*, *sentPapers* and *acceptedPapers* have been instantiated in the process with the values {500, 750, 2500, 8500, 150} respectively, in the activity *Print Proceedings* the interval {[67, 123]} is obtained in order to ascertain

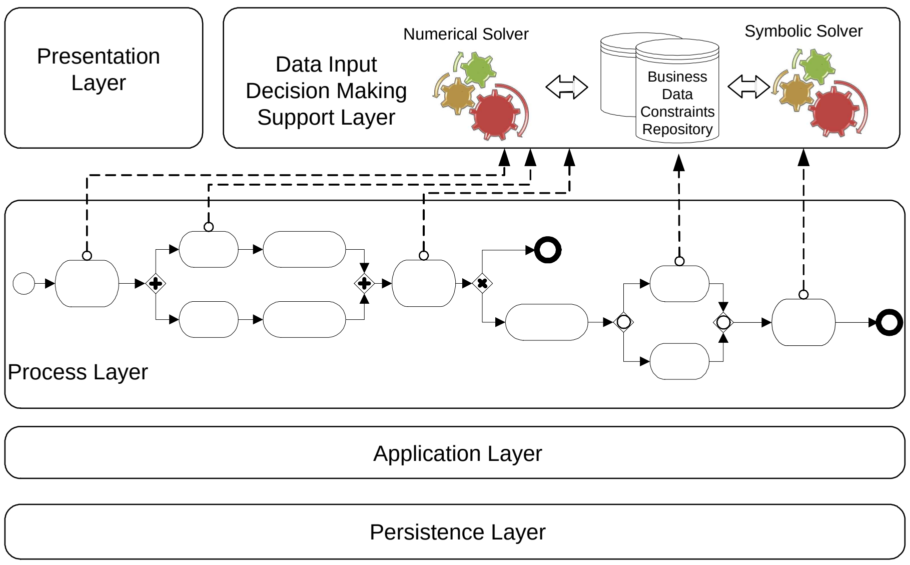


Fig. 7. DSS for Decision-making support for Input Data

the possible values of the decision variable *numberOfProceedings*.

7. Implementation details in a Case of Study

In order to show the benefits of the decision-making support for input data in business process instances, we have implemented a solution based on the Process Aware Information System (PAIS) framework and illustrated how to facilitate the input data support into a commercial solution.

7.1. *Decision Support System for Input Data in Business Processes*

In order to permit the decision-making support for input data at various points of the process based on the BDCs, this paper is based on an extension of the classic PAIS framework [16], as presented in [17], and shown in Figure 7. In general, a PAIS architecture [18] can be viewed as a 4-tier system as presented in [16], 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.

Data decision-making support and business process layers are two parallel and "independent" systems. They are considered independent since they can be simultaneously executed in separate machines, for different applications. However, this independence fails from the point of view of dataflow information, since, for the various decision points, the Data Input Decision layer uses the instantiated variables in the dataflow and uses the decision variables that the process needs. With the presented DSS, it is possible to design both the business process model and the BDCs, thereby achieving higher levels of flexibility and agility in the business process management. One of the items that also needs to be studied is how to store the BDCs, and how to define the relation between each of them and the activities of the process. When a great deal of BDCs have to be handled, the use of a database to store and manage these constraints is mandatory, especially when not all the BDCs are established for the whole business process, and the relation between activities and BDCs has to be defined.

The necessity to store the business compliance rules was analysed in [19], but failed to take data semantics into account. However, BDCs cannot be stored in a classic relational database, since storing a BDC also implies storing all the details related to its variables, the domain of variables, and data persistence relationships. The difficulty in storing BDCs arises due to the problem of how to store the constraints themselves as data, since they do not belong to a type supported by commercial databases. In order to manage constraints, we propose the use of Constraint Database Management Systems (CDBMS) as explained in [20]. That proposal is based on an envelope for a database management system to manage Constraints as a classic type, which has been proposed for the description of the BDCs. A similar way to store BDCs was used in previous work [7]. This solution shields the user from unnecessary details on how the BDCs are stored and queried.

Once how to store the BDCs is ascertained, the next question is how each BDC is related to each activity and with the rest of the process. Figure 8 represents the relations necessary to describe that a *Process* has a set of *Dataflow variables*, available for the *Activities*. Each *BusinessDataConstraint* can be associated with a set of activities or to the whole process, while each Activity can have several associated BDCs. These associations are established in the table *Activity/BDC*.

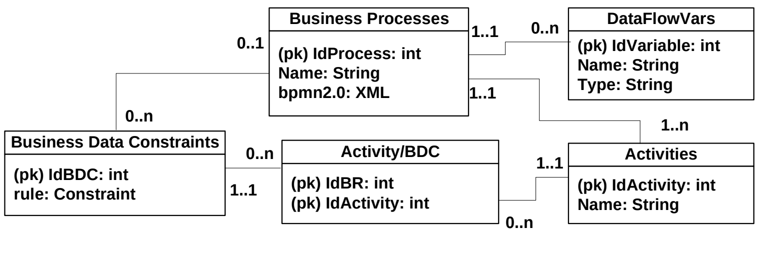


Fig. 8. Relations between BDCs and Activities

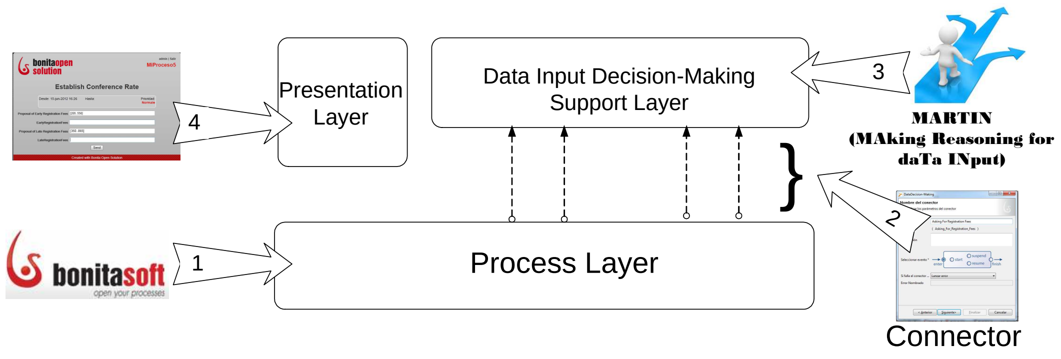

Presentation Layer
Data Input Decision-Making Support Layer
Process Layer
1
2
3
4
MARTIN (MAking Reasoning for daTa INput)
Connector
Establish Conference Rate

Fig. 9. DSS for the decision-making support for input data with tools

7.2. *Computational application for the Decision-Making support for Input Data*

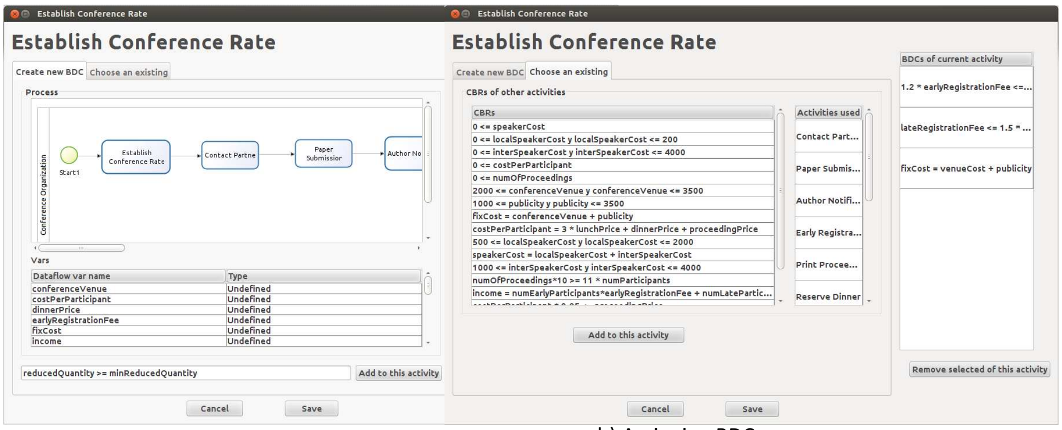
In order to facilitate the creation of BDCs and the decision-making of the process, we have implemented an application and a connector that follow the DSS presented in Section 7.1. This solution uses a specific set of technologies that could be replaced by another set. The specific configuration that we have implemented is presented in Figure 9, which describes a possible combination of tools to execute the decision-making support for input data. We have prepared a video [21] where the steps for the design and execution of the decision support are shown. The steps to configure and use the application are:

(1) **Modelling the business process and defining the dataflow variables ($\mathbb{P}$, $\mathbb{DF}$):** The business process and the dataflow variables can be modelled in any Business Process Management System, for example: Intalio^{TM}, Activiti^{TM}, and Bonita Open Solution^{TM}. In the case of study presented, we have used Bonita Open Solution^{TM} since it is an open-code application with free distribution, and is commonly used in the private company sector. Once the process is modelled, the designer of the process must decide on the decision points associated with any activity.

(2) **Locating the decision points and decision variables ($\mathbb{DV}$):** If a designer considers allocating a decision point into a determined activity, then a connector must be added in the activity to relate it with the software that executes the decision. We have implemented a connector (as shown in the video [21]) to facilitate the relation between the decision point and the reasoning software (MARTIN) that obtains the possible and correct values of the decision variables. When a connector is included in an activity, it is necessary to define the decision variables. In the connector, called "Data Decision-Making Connector", all the dataflow values and the decision data variables are sent at runtime to

the software that executes the evaluation of the decision-making support for the model obtained at design time, as explained in Section 6.

(3) **Creating the BDCs ($\mathbb{BDC}$):** Once the process model is defined, it is necessary to create the BDCs and associate them to each activity, using the solution of Figure 8. The decision-making process uses all the variables instantiated in the dataflow at runtime, and the BDCs that represent the possible correct values in the future. To this end, we have implemented an application called *MARTIN: MAking Reasoning for daTa INput*, to facilitate the creation of BDCs, and the association of the BDCs to the activities. Most commercial tools provide an XML representation of the created process that follows the BPMN 2.0 [13]. For this reason, we have implemented a transformation from a *.bpmn* file that represents the type models described in this paper (formed by activities and control flows), into our *Process Graph*. When the XML of the process is analysed, the tables *Business Process*, *Activities*, and *DataflowVar* shown in Figure 8 are filled in automatically, since the XML of the process holds all this information. It is then possible to create and/or assign BDCs to the various activities. In order to provide a simple way for the business expert to add BDCs, the interface (shown in Figure 10.a) enables the business process model to be viewed, and BDCs to be assigned to the activities that belong to the business process (Figure 10.b).



Dataflow var name	Type
conferenceVenue	Undefined
costPerParticipant	Undefined
dinnerPrice	Undefined
earlyRegistrationFee	Undefined
fixCost	Undefined
income	Undefined

reducedQuantity >= minReducedQuantity

CBRs
0 <= speakerCost
0 <= localSpeakerCost y localSpeakerCost <= 200
0 <= interSpeakerCost y interSpeakerCost <= 4000
0 <= costPerParticipant
0 <= numOfProceedings
2000 <= conferenceVenue y conferenceVenue <= 3500
1000 <= publicity y publicity <= 3500
fixCost = conferenceVenue + publicity
costPerParticipant = 3 * lunchPrice + dinnerPrice + proceedingPrice
500 <= localSpeakerCost y localSpeakerCost <= 2000
speakerCost = localSpeakerCost + interSpeakerCost
1000 <= interSpeakerCost y interSpeakerCost <= 4000
numOfProceedings*10 >= 11 * numParticipants
income = numEarlyParticipants*earlyRegistrationFee + numLatePartic...

Activities used
Contact Part...
Paper Submis...
Author Notifi...
Early Registra...
Print Procee...
Reserve Dinner

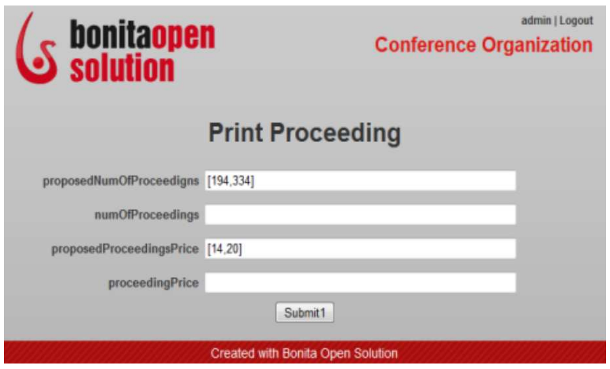
BDCs of current activity
1.2 * earlyRegistrationFee <=...
lateRegistrationFee <= 1.5 * ...
fixCost = venueCost + publicity

a) Creating BDCs
b) Assigning BDCs

Fig. 10. Connector to create and assign BDCs to activities

(4) **Instantiating the business process model ($\mathbb{DFI}$) and obtaining the decision variables instantiation ($\mathbb{DVI}$).** Once the business process model has been designed in a Business Process Management System, the execution process can be performed. When a business process instance executes an activity with a *Data Decision-Making Connector*, the dataflow values of that instance

and data decision are sent to the Data Input Decision-Making Support layer which implements the evaluation of models, as explained in Section 6. Figure 11 presents an example of the form that shows the possible intervals for each decision variable.



Print Proceeding

proposedNumOfProceedigns	[194,334]
numOfProceedings	
proposedProceedingsPrice	[14,20]
proceedingPrice	

Fig. 11. Form of Activity *Establish Conference Rate* at runtime

8. Related Work

Decision-making support in business processes carries significant contributions related to how to model the process, which in turn help the designer to decide the best combination of activities to achieve an objective. A simulation-based approach to decision-making support is proposed in [22] with respect to complex dynamic systems, and includes uncertain data. A methodology to optimize a process where the description is not clear (fuzzy) is put forward in [23]. However, in both papers, the help in the business process has been oriented towards the design of the model, or the redesign of the business process [24], by looking at the quality of the process at design time, but not how this process works at runtime, thereby missing the importance of the variables of the dataflow. Data has also been involved in other studies related to decision support; for example [25] proposes an operational decision support for the construction of process models based on historical data to simulate processes. That proposal includes a generic approach to a business process for operational decision support, and includes business process modelling and workflow simulation with the models generated, by using process mining. Other work related to how to model the processes, such as that in [26], proposes a framework of assistance to create models which take the necessary resources involved in the process into account. In that paper, the data that describe the resources of the execution of the process are used, but not the data that flows at runtime, nor does it consider how this assistance can help at runtime.

In general, papers found in the literature related to decision-making support are not focused on the assistance of the user for the input data. Although work such as [27] is oriented towards auditing the process in order to detect gaps between the information system process flow and the internal control flow in the business process, the quality of the data values at runtime is not a cause of concern for the authors. This detection can be derived from the existence of an oversight in the description of the semantics of data in the business processes. The use of compliance rules has traditionally been used for the validation of the business process, not for user assistance. The validation of business process traces has been a field of intense research over recent years using business compliance rules: see [28] as an entry point into this literature. However, these types of proposal cannot be used in the decision-making support for input data, since they are focused on the compliance of the process model structure [29], [30].

Regarding how to model data-aware compliance rules, studies such as [10], [31], [32], and [34], have defined graphical notations to represent the relationship between data and compliance rules by means of data conditions. These types of compliance rules cannot therefore be used to infer the possible values of the variables that are involved in the decisions. In [35], "semantic constraints" and the SeaFlows framework for enabling integrated compliance support are proposed. Furthermore, in [37], a preprocessing step to enable data-aware compliance checking in an efficient manner is presented: the data describe under what conditions the activities can be executed. In general, many examples can be found where data objects are used for compliance verification, for instance, the semantically annotating activities with preconditions and effects that may refer to data objects are introduced in [38], but none of these factors assist the user with this information at runtime.

Summarizing, to the best of our knowledge, only one preliminary study [17] exists that uses the knowledge of the business process model and the BDCs for decision-making support for input data, while the rest of the proposals are focused on the design or re-design of the business process model. The current paper constitutes an improvement on the previous paper by: including the process graph definition to represent the model; defining the algorithm to traverse the graph; and by implementing an application that can be integrated with commercial tools.

9. Conclusions and Future work

In this paper, the use of BDCs is proposed to infer the possible input data values in a business process instance, by taking into account the model of the process and the decision points where the assistance must be executed. In order to meet this challenge, two different parts have been distinguished: an analysis of the process model at design-time, by means of an algorithm to traverse the business process model; and an evaluation of the decision model at runtime, to obtain the possible values of the decision variables using constraint programming. In order to implement this proposal, a case of study has been developed using Bonita Soft and an application called MARTIN has been included in order to create, include, and evaluate the

BDCs in the business process model.

One aspect that can be improved in this work is the presentation of the possible and correct values of the decision variables to the user. In the case of numerical decision support, this improvement could be attained by showing the user the possible combinations for various decision variables simultaneously, instead of solely presenting the intervals for each variable. As future work, we also plan to include a symbolic representation of the possible correct variables instead of interval values.

Further research could be analysed in greater depth, such as: a) In the current proposal, all the BDCs are defined for all instances, but it would be possible, depending on the values of the dataflow or the moment where the instance is executed, that a different set of BCDs could be involved in the decision support. b) Related to the above proposal, there is also the possibility of including new BDCs for the user at runtime, which would render the decision-making support for input data more customized. This would help the user to include, for example, reductions of the domains for determined decision variables associated with future decision points, even before the final value is introduced. If, at a decision point, there are no possible values for a decision variable, it implies that no BDC is consistent. For future work, we propose reinforcing the detection of the minimum set of BDCs that are not satisfiable, and ascertaining how this situation can be fixed by means of input data decisions.

Acknowledgement

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).

Appendix

Dataflow Variables:

- `totalCost`, `income` is the total cost and income of the conference respectively
- `numEarlyParticipants` is the number of people attending, registered in the early period
- `numLateParticipants` is the number of people attending, registered in the late period
- `numParticipants` is the final number of participants
- `costPerParticipant` is the cost per participant, and includes lunch and dinner, cost of proceedings, etc.
- `earlyRegistrationFee` is the cost of early registration
- `lateRegistrationFee` is the cost of late registration
- `sponsorship` is the income from the companies that sponsor the event
- `dinnerPrice` is the price of the gala dinner

- `lunchPrice` is the price of the lunch
- `proceedingsPrice` is the price of printing one copy of the proceedings
- `numOfProceedings` is the number of proceedings printed
- `acceptedPapers` is the number of accepted papers
- `sentPapers` is the number of sent papers
- `speakerCost` is the cost of inviting a speaker to the conference, local or international
- `localSpeakerCost` is the cost of inviting a local speaker to the conference
- `interSpeakerCost` is the cost of inviting an international speaker to the conference
- `venueCost` is the cost of booking the place to hold the conference
- `publicity` is the cost of publicizing the conference
- `fixCost` is the fixed cost independent of the number of attending people, the venue and the publicity cost

Range of the Dataflow Variables and Business Data Constraints for the whole process:

- $\{5000 \leq \text{totalCost} \leq 50000\}$
- $\{5000 \leq \text{income} \leq 60000\}$
- $\{50 \leq \text{numEarlyParticipants} \leq 200\}$
- $\{10 \leq \text{numLateParticipants} \leq 100\}$
- $\{60 \leq \text{numParticipants} \leq 300\}$
- $\{0 \leq \text{costPerParticipant} \leq 900\}$
- $\{120 \leq \text{earlyRegistrationFee} \leq 1000\}$
- $\{120 \leq \text{lateRegistrationFee} \leq 1000\}$
- $\{1000 \leq \text{sponsorship} \leq 10000\}$
- $\{30 \leq \text{lunchPrice} \leq 100\}$
- $\{60 \leq \text{dinnerPrice} \leq 250\}$
- $\{5 \leq \text{proceedingsPrice} \leq 20\}$
- $\{10 \leq \text{numOfProceedings} \leq 400\}$
- $\{0 \leq \text{acceptedPapers} \leq +\infty\}$
- $\{0 \leq \text{sentPapers} \leq +\infty\}$
- $\{0 \leq \text{speakerCost} \leq 7000\}$
- $\{0 \leq \text{localSpeakerCost} \leq 2000\}$
- $\{0 \leq \text{interSpeakerCost} \leq 4000\}$
- $\{2000 \leq \text{venueCost} \leq 3500\}$
- $\{1000 \leq \text{publicity} \leq 2500\}$
- $\{0 \leq \text{fixCost} \leq 9000\}$
- $\{\text{speakerCost} = \text{localSpeakerCost} + \text{interSpeakerCost}\}$
- $\{\text{costPerParticipant}=3*\text{lunchPrice}+\text{dinnerPrice}+\text{proceedingsPrice}\}$

Business Data Constraints for activities:

- *Establish Conference Rate*

  - {1.2*earlyRegistrationFee $\leq$ lateRegistrationFees $\leq$ 1.5*earlyRegistrationFee}
  - {fixCost = venueCost+publicity}
- *Paper Submission*
  - {sentPaper*0.3 $\leq$ acceptedPapers $\leq$ sentPaper*0.8}
- *Author Notification*
  - {numEarlyParticipant*0.8 $\leq$ acceptedPapers $\leq$ numEarlyParticipant*1.2}
- *Print proceedings*
  - {costPerParticipant*0.05 $\leq$ proceedingsPrice $\leq$ costPerParticipant*0.15}
  - {numOfProceedings $\geq$ 1.1 * numberOfParticipant}
- *Early Registration*
  - {1.2*numEarlyParticipants $\leq$ numParticipant $\leq$ 1.4*numEarlyParticipants}
- *Book Lunch*
  - {costPerParticipant*0.10 $\leq$ lunchPrice*3 $\leq$ costPerParticipant*0.35}
- *Book Dinner*
  - {costPerParticipant*0.10 $\leq$ dinnerPrice $\leq$ costPerParticipant*0.60}
- *Invite Local Speaker*
  - {sponsorship*0.3 $\leq$ localSpeakerCost $\leq$ sponsorship*0.5}
- *Invite International Speaker*
  - {sponsorship*0.3 $\leq$ interSpeakerCost $\leq$ sponsorship*0.5}
- *Hold Conference*
  - {totalCost = numParticipant$*$costPerParticipant + inviteSpeaker +numOfProceedings$*$proceedingsPrice + fixCost}
  - {income*0.7 $\leq$ totalCost $\leq$ income*0.9}
- *Final Report & Make Payment*
  - {numParticipants = numEarlyParticipants + numLateParticipants}
  - {Income = numEarlyParticipants $*$ earlyRegistrationFee + numLateParticipants $*$ lateRegistrationFee + sponsorship}

References

1. M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
2. 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.

3. 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.
4. R. G. Ross, What is a business rule? Practicable business rules, Business.
5. N. Sponsor, Business rules and business processes, Information Systems Journal 1 (10) (2008) 20–24.
6. J. Becker, C. Ahrendt, A. Coners, B. Wei, A. Winkelmann, Modelling and analysis of business process compliance., M. Nttgens, A. Gadatsch, K. Kautz, I. Schirmer, N. Blinn (Eds.), Governance and Sustainability in Information Systems, Vol. 366 of IFIP Publications, Springer, 2011, pp. 259–269.
7. 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.
8. R. Wörzberger, T. Kurpick, T. Heer, Checking correctness and compliance of integrated process models, in: SYNASC, 2008, pp. 576–583.
9. M. T. Gómez-López, R. M. Gasca, A. Reina-Quintero, Model-driven engineering for constraint database query evaluation, in: Workshop Model-Driven Engineering, Logic and Optimization: friends or foes?, MELO 2011, Springer, 2011, pp. 5–20.
10. I. Weber, J. Hoffmann, J. Mendling, Semantic business process validation, in: SBPM'08: 3rd international workshop on Semantic Business Process Management at ESWC'08, 2008.
11. R. Dechter, Constraint Processing (The Morgan Kaufmann Series in Artificial Intelligence), Morgan Kaufmann, 2003.
12. G. Rochart, N. Jussien, X. Lorca, Choco. A java constraint programming library, Reference Manual. http://www.emn.fr/z-info/choco-solver/.
13. OMG. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0. January 2011. http://www.omg.org/spec/BPMN/2.0/.
14. S. Basu, Algorithms in semi-algebraic geometry, ph D thesis (1996).
15. W. Hodges, Some Strange Quantifiers. Structures in Logic and Computer Science, 1997, pp. 51–65.
16. B. Weber, S. W. Sadiq, M. Reichert, Beyond rigidity - dynamic process lifecycle support, Computer Science - R&D 23 (2) (2009) 47–65.
17. 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: International Conference on Information System Development, ISD 2011, Springer, 2011, pp. 15–25.
18. 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.
19. S. Rinderle-Ma, S. Kabicher, L. T. Ly, Activity-oriented clustering techniques in large process and compliance rule repositories, in: Business Process Management Workshops (2), 2011, pp. 14–25.
20. 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.
21. M. T. Gómez-López, R. M. Gasca, J. M. Pérez-Álvarez, Decision support system for input data in business processes, `http://www.lsi.us.es/~quivir/mayte/martin.html`
22. P. Volkner, B. Werners, A decision support system for business process planning, European Journal Of Operational Research 125 (3) (2000) 633–647.
23. P. Vlkner, B. Werners, A simulation-based decision support system for business pro-

cess planning. Fuzzy Sets and Systems 125 (3) (2002) 275–287.

24. N. Kock, J. Verville, A. Danesh-Pajou, D. DeLuca, Communication flow orientation in business process modeling and its effect on redesign success: Results from a field study, Decision Support Systems 46 (2) (2009) 562–575.
25. Y. Liu, H. Zhang, C. Li, R. J. Jiao, Workflow simulation for operational decision support using event graph through process mining, Decis. Support Syst. 52 (3) (2012) 685–697.
26. I. Barba, B. Weber, C. D. Valle, Supporting the optimized execution of business processes through recommendations, in: Business Process Management Workshops (1), 2011, pp. 135–140.
27. S.-M. Huang, D. C. Yen, Y.-C. Hung, Y.-J. Zhou, J.-S. Hua, A business process gap detecting mechanism between information system process flow and internal control flow, Decis. Support Syst. 47 (4) (2009) 436–454.
28. 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.
29. S. W. Sadiq, M. E. Orlowska, W. Sadiq, Specification and validation of process constraints for flexible workflows, Inf. Syst. 30 (5) (2005) 349–378.
30. 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.
31. 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.
32. J. Hoffmann, I. Weber, G. Governatori, On compliance checking for clausal constraints in annotated process models, in: Information Systems Frontiers, Vol. 14, Num. 2, 2012, pp. 155–177.
33. M. zur Muehlen, M. Indulska, Modeling languages for business processes and business rules: A representational analysis, in: Information Systems, Vol. 35, Num. 4, 2010, pp.379–390.
34. 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.
35. L. T. Ly, S. Rinderle-Ma, K. Göser, P. Dadam, On enabling integrated process compliance with semantic constraints in process management systems, Information Systems Frontiers (2009) 1–25.
36. S. Ermon, Y. Xue, J. Conrad, C. Gomes, B. Selman, Combinatorial Decision Making in Complex, Uncertain, and Highly Dynamic Environments, 3rd International Conference on Computational Sustainability (2012).
37. 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.
38. G. Governatori, J. Hoffmann, S. W. Sadiq, I. Weber, Detecting regulatory compliance for business process models through semantic annotations, in: Business Process Management Workshops, Vol. 17 of Lecture Notes in Business Information Processing, Springer, 2008, pp. 5–17.