Hybrid business process modeling for the optimization of outcome data

Luisa Parody*, María Teresa Gómez-López, Rafael M. Gasca

Department of Languages and Computer Systems, University of Seville, Spain

Keywords:
Hybrid model
Business process
Constraint programming
Data optimization

A B S T R A C T

Context: Declarative business processes are commonly used to describe permitted and prohibited actions in a business process. However, most current proposals of declarative languages fail in three aspects: (1) they tend to be oriented only towards the execution order of the activities; (2) the optimization is oriented only towards the minimization of the execution time or the resources used in the business process; and (3) there is an absence of capacity of execution of declarative models in commercial Business Process Management Systems.

Objective: This contribution aims at taking into account these three aspects, by means of: (1) the formalization of a hybrid model oriented towards obtaining the outcome data optimization by combining a data-oriented declarative specification and a control-flow-oriented imperative specification; and (2) the automatic creation from this hybrid model to an imperative model that is executable in a standard Business Process Management System.

Method: An approach, based on the definition of a hybrid business process, which uses a constraint programming paradigm, is presented. This approach enables the optimized outcome data to be obtained at runtime for the various instances.

Results: A language capable of defining a hybrid model is provided, and applied to a case study. Likewise, the automatic creation of an executable constraint satisfaction problem is addressed, whose resolution allows us to attain the optimized outcome data. A brief computational study is also shown.

Conclusion: A hybrid business process is defined for the specification of the relationships between declarative data and control-flow imperative components of a business process. In addition, the way in which this hybrid model automatically creates an entirely imperative model at design time is also defined. The resulting imperative model, executable in any commercial Business Process Management System, can obtain, at execution time, the optimized outcome data of the process.

1. Introduction

A business process, henceforth referred to as BP, consists of a set of activities that are executed in coordination within an organizational and technical environment. These activities jointly attain a business goal [73]. Several languages propose an imperative representation of business processes. An imperative specification allows business ex-perts to describe an explicit order of execution between activities, and to transform the process into an executable model [1]. Therefore, an imperative description defines exactly how the activities have to be performed, and how the data-flow should be handled (for exam-ple, that activities A, B, and C are executed sequentially, or activities D and E are executed in parallel). However, the knowledge about the systems can sometimes be described by means of the things that are permitted or prohibited.

Declarative descriptions enable specification of *what* has to be done, instead of *how* it has to be done (for example, activity A cannot be executed before activity B ends). Although imperative models are significantly more understandable than declarative models [14,15,54], declarative specifications can complement an imperative model when an imperative description cannot be performed. This is the reason why se veral authors have proposed languages for the definition of BPs as declarative models [33,43,51,60,63]. In addition, this modeling can be used when the BP cannot always be defined prior to execution time. Indeed, imperative models should be completely known and specified at design time, since they explicitly represent all the allowed sequences of activities. On the other hand, declarative models describe which orders of activities are permitted in an open world assumption (everything that is not explicitly specified is permitted); for example, if activity *A* is executed, then activity *B* has

to be executed afterwards, and, as long as this rule is satisfied, any behavior is allowed. In this way, at design time, the BP can remain "underspecified".

Unfortunately, there remain three main aspects of these declarative languages that need major improvement: (i) the capacity of the data-oriented description; (ii) the management of the data that optimizes the outcome data of each instance at runtime; and (iii) the necessity to make the declarative models executable and integrate them into a Business Process Management System. Each of these aspects is detailed below:

(i) Data-oriented in declarative models: There is a significant number of papers that detect the necessity to include the data relationships description into the BP model. Although certain imperative languages, such as BPMN 2.0 [48], have included components to describe the data exchange during the execution of the process, new extensions have been proposed because the standard has insufficient descriptive capacity, such as in [23,41,42]. The role of data in declarative languages has been oriented to describe how the values of the data at runtime can affect to the activities execution order [4,32,36]. In this paper, the objective at runtime is to ascertain the values of the data to optimize the product obtained from the BP.

(ii) Outcome optimization: Since the declarative models provide an underspecified definition of business process requirements at design time, the activities can be executed in different orders at runtime, but still in accordance with these requirements. Due to this fact, declarative models are typically joined to optimization problems. In general, this optimization is oriented towards minimizing the execution time, or reducing the resources used in the BP [26], but not to the optimization of the outcome data of each instance. One of the definitions of BP is *"a series of steps designed to produce a product or service"* [58]. Therefore, the optimization of the process to obtain the product, or to obtain a better product, is the reason why a company becomes more competitive than others. Therefore, the objective of a company can be oriented towards decreasing the cost of development time, or obtaining the best product on the market in order to be more competitive. Our proposal is focused on this latter group, specifically, in the optimization of the outcome data by mean of supporting customers about the best input data for each instance.

(iii) Executable declarative models: Declarative languages enable a description of the temporal order of the activities to be written, and the necessary resources to perform them. However, although several tools are available to execute declarative descriptions, none of them provides anything more than guidelines or recommendations about which activity should be executed at a specific point of an instance. Unfortunately, there are no Business Process Management Systems (BPMS) available for the execution of declarative models in the same way as there are for imperative models. For example, they do not support users in choosing the best input data for each instance, in order to optimize the outcome data of the process. Therefore, declarative description are not used in the daily work of companies.

1.1. Detailing a case study: trip planner

In order to understand the weakness of the existing declarative proposals (see Section 5), we introduce an example related to a trip planner. Our proposal can be applied to any example where the objective of a BP is to obtain the data corresponding to the best product (outcome data of the process) based on the customer requirements. The difficulty is that each customer can have different requirements, and the same BP model needs to satisfy the necessities, being flexible in this aspect. The example pertains to a trip planner process, based on that presented in [49] and [50]. The process describes the activities for the booking of flights, reservation of a hotel room, and, if necessary, the renting of a car. In order to minimize the price, the customer can choose among different dates, or can change the airport by traveling to a nearby city with a rented car. For this process, the model is introduced, thereby making it possible to execute the activities in a parallel manner. The problem now is to determine the best combination of data input for the activities in order to minimize the total price.

In order to create a workflow process where the activities are involved, the business process shown in Fig. 1 can be modeled with the standard Business Process Model and Notation (BPMN) [48]. The process starts with (i) the travel reservation request; follows with (ii) the searching of flights, hotel rooms, and rental car that compose the cheapest travel package that fits customer preferences; (iii) the travel package is offered to the customer; and finally, (iv) the customer, depending on his/her preferences, either formalizes the proposed travel package or cancels it. The problem in this model lies in part (ii), where the search for each component of the trip is performed. This search is in accordance with the availability and the customer requirements: the place to visit, the possible dates, the price, etc.

It is possible to combine the activities that represent the three providers (Hotel, Flight, and Car Rental Provider) into a single BP, where the activities are executed in parallel. The question becomes how to determine that the best trip is found (the business product obtained as data output), and which BP model minimizes a value determined by a function in accordance with the outcome data (the total price in this example). Unfortunately, the trip planner problem cannot be modeled using the languages found in the literature, since it is not possible to describe: (i) the unknown input values of the data of the activities; (ii) the objective function according to the data output of the activities; and (iii) how the input values in an activity can affect other activities.

1.2. Our proposal

In order to solve the aforementioned limitations, the input data dependencies are dealt with by means of a data-oriented optimization problem. The proposal consists of two parts:

(i) *Formalization of a hybrid model to represent the outcome data optimization:* The Data-Oriented OPTimization DEClarative language, called DOOPT-DEC, is introduced to formalize a model that includes the process requirements referring to the data description in a declarative way. This description can be included in an imperative model that represents the control-flow requirements. More specifically, these requirements are necessary when the input data of each activity is unknown at design time, and need to be described in a declarative way to be discovered for each instance at runtime. In this paper, a new point of view of declarative languages focused on data is presented.

(ii) *Creation of an imperative model from a declarative description:* Imperative specification implies "saying how to do something", whereas declarative specification supposes "saying what is required and letting the system determine how to achieve it". The proposal in this paper is focused on building an imperative model which obtains the best combination of data from the activities so that the optimal outcome data is attained, while maintaining the capacities of the declarative description thanks to the use of Constraint Programming in a BPMS.

The remainder of the paper is organized as follows: Section 2 introduces the proposed declarative language with data aspects for use in the hybrid model. The possible models that can be created are detailed in Section 3. In addition, Section 3 also explains how this hybrid model can be transformed into an imperative computable model, and how the optimization problem can be solved using the Constraint Programming paradigm. Section 4 includes a brief computational and statistical study of our proposal. Section 5 includes certain related

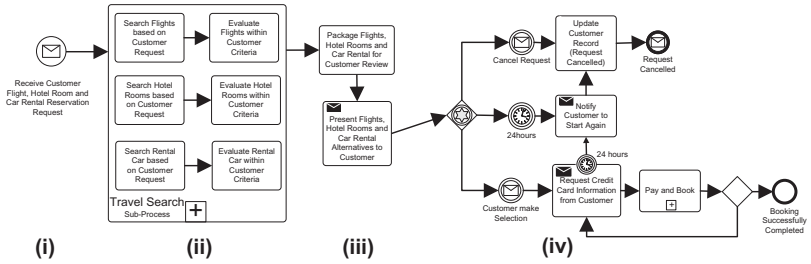

Fig. 1. Example of trip planner [49].

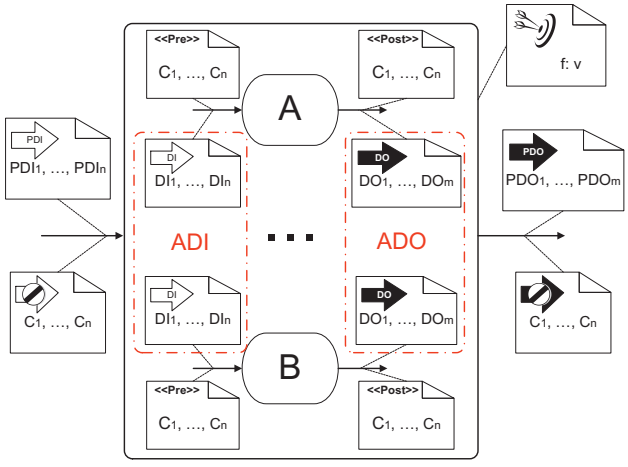

Fig. 2. Subprocess components.

work and its comparison with our proposal. Finally, conclusions are drawn and future work is proposed in Section 6.

2. Formalization of the data-oriented optimization business process languages

In this section, the model is formalized and a language with graphical notation, called DOOPT-DEC, is provided to facilitate the description of the data-oriented optimization declarative models. The purpose of the DOOPT-DEC language is to enrich the imperative model by adding the declarative description of the relationships between data. Therefore, the division of the formalization is aligned to the relation between the imperative part of the model (subprocess description) and the declarative specification (subprocess relationships), as shown in Fig. 2. Using this hybrid description, another model exclusively imperative is created by including a constraint programming task to determine at runtime the most appropriate data input to optimize the outcome of the process.

In order to introduce the formalization of the data provided and combined (see Fig. 2), the different descriptions to include are divided into: (i) subprocess description, including the descriptions of the components associated with the activities of the imperative part, detailed in Section 2.1; and (ii) subprocess relationships, which is the declarative description of the relationships between the components through the data-flow ($\mathbb{DF}$), addressed in Section 2.2.

2.1. Subprocess description

The *subprocess description* includes the components associated with the activities, gateways, and control-flow which are known and can be represented in an imperative way. Taking $\mathbb{A}$ as the finite set of activities $\{A_1, \ldots, A_i, \ldots, A_n\}$ contained in a determined BP, the following definitions are introduced.

Definition 2.1. $\mathbb{ACTIVITIES\_DATA\_INPUT}$ ($\mathbb{ADI}$) and $\mathbb{ACTIVITIES\_DATA\_OUTPUT}$ ($\mathbb{ADO}$) represent the sets containing, respectively, all the data input and data output involved in the execution of all the activities.

Within the set $\mathbb{ADI}$ ($\mathbb{ADO}$), containing all the input (output) variables, it is possible to identify the input (output) variables of each activity, defined as follows.

Definition 2.2. $\mathbb{DATA\_INPUT}(A_i)$ and $\mathbb{DATA\_OUTPUT}(A_i)$ describe the set of data input and output of an activity A_i involved in the subprocess, respectively, so that

$\forall A_i, \mathbb{DATA\_INPUT}(A_i) \subseteq \mathbb{ADI}$, where $\mathbb{ADI} \subset \mathbb{DF}$.

And, $\forall A_i, \mathbb{DATA\_OUTPUT}(A_i) \subseteq \mathbb{ADO}$, where $\mathbb{ADO} \subset \mathbb{DF}$.

Since various activities can share the same data input (or output), it is also possible that:

$\mathbb{DATA\_INPUT}(A_i) \cap \mathbb{DATA\_INPUT}(A_j) \neq \emptyset$, for i ≠ j.

And, $\mathbb{DATA\_OUTPUT}(A_i) \cap \mathbb{DATA\_OUTPUT}(A_j) \neq \emptyset$, for i ≠ j.

The unions of all the $\mathbb{DATA\_INPUT}$ sets and all the $\mathbb{DATA\_OUTPUT}$ sets of all the activities, constitute the sets $\mathbb{ADI}$ and $\mathbb{ADO}$ respectively, with non-repetitive elements.

$\mathbb{ADI} = \{\mathbb{DATA\_INPUT}(A_1) \cup \ldots \cup \mathbb{DATA\_INPUT}(A_n)\}$

$\mathbb{ADO} = \{\mathbb{DATA\_OUTPUT}(A_1) \cup \ldots \cup \mathbb{DATA\_OUTPUT}(A_n)\}$

Likewise, the specific variables that are inputs or outputs of the overall subprocess can be distinguished. They represent the information that flows from the customer to the subprocess and vice versa:

Definition 2.3. $\mathbb{PROCESS\_DATA\_INPUT}$ ($\mathbb{PDI}$) is the set of input variables of the subprocess, which determines the information provided and defined by the customer. $\mathbb{PDI}$ is composed of a subset of variables of $\mathbb{ADI}$.

$\mathbb{PDI} \subseteq \mathbb{ADI}$

Definition 2.4. $\mathbb{PROCESS\_DATA\_OUTPUT}$ ($\mathbb{PDO}$) is the set of output variables of the subprocess. $\mathbb{PDO}$ is composed of a subset of variables of $\mathbb{ADO}$.

2.2. Subprocess relationships

There exist different relationships between the data inputs and data outputs defined in the *subprocess description*. Those relationships are expressed as constraints, both at activity and at process level, giving rise to the *subprocess relationships*, with the following definitions:

Definition 2.5. $\mathbb{PRE}(A_i)$ is the set of constraints that limits the specific values of the $\mathbb{DATA\_INPUT}(A_i)$ that must be satisfied to execute activity A_i. Likewise, $\mathbb{POST}(A_i)$ is the set of constraints that limits the

specific values of the DATA_OUTPUT(A_i) that must be satisfied after the execution of activity A_i.

Definition 2.6. INPUT_CONSTRAINT (OUTPUT_CONSTRAINT) relates the values of variables of PDI (PDO) with variables of DATA_INPUT (DATA_OUTPUT) of each activity. These constraints limit the possible value that the DATA_INPUT (DATA_OUTPUT) can take according to the PDI (PDO) values in each instance, and the possible values between the input and output of the activities.

Definition 2.7. OBJECTIVE_FUNCTION (OBJ_FUNC), is an optimization function defined in terms of the data output of the activities (ADO).

The objective of this optimization function is either to maximize or to minimize some of the output data that represent the business product, which constitutes the outcome of the process.

$\mathbb{OBJ\_FUNC} : f(v \subseteq \mathbb{ADO}) \rightarrow value$, where f is *MAX* or *MIN*

The result of this optimization problem is a set of input values that satisfies the optimization function, the pre and post-conditions of the activities, and also satisfies the input and output constraints.

The set of input values that optimizes the output is found at runtime. The role of the constraints is to determine the possible values that this input and output data can take. All the constraints (INPUT_CONSTRAINTS, OUTPUT_CONS– TRAINTS, PRE and POST) are always defined at design time by a business expert who is familiar with the problem, although they are solved for each instance at runtime. Among every possible tuples of solutions that satisfy the constraints, the outcome of the subprocess will be the one which optimizes the variable defined in the optimization function.

2.3. Grammar and graphical notation

The formalization presented above, used as a data-oriented optimization process, includes: the data of the process, the data of the activities, the objective function, and the constraints. These constraints are defined by the following grammar, where `Variable` and `Constant`, can be defined in the Integer, Natural, Float, Dates, and String domains. On the other hand, `Set` is defined as a set of `Constant` values of a specific `Variable`.

```
   Constraint   :=   Atomic_Constraint BOOL_OP
Constraint
  |Atomic_Constraint
  |'¬' Constraint
  |Variable SET_FUNCTION Set
  BOOL_OP := '∨'|'∧'|'→'
  SET_FUNCTION := '∈'|'∉'
  Atomic_Constraint   :=   function PREDICATE
function
  function := Variable FUNCTION_SYMBOL function
  |Variable
  |Constant
  PREDICATE := '='|'≠'|'<'|'≤'|'>'|'≥'
  {For the String domain only '=' and '≠' '
are allowed}
  FUNCTION_SYMBOL := '+'|'−'|'*'|'/'
  {These operators are only applicable to
Numerical variables}
```

A graphical notation, called DOOPT-DEC (Data-Oriented OPTimization DEClarative language), is defined in order to include these components into a BP model easily.

DOOPT-DEC enables an imperative description focused on the control-flow perspective, using BPMN, to be combined with a declarative description of the data perspective of the subprocess. The existing BPMN components are combined with the new data components

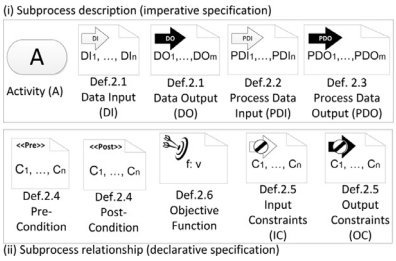

Fig. 3. DOOPT-DEC Graphical Components.

in order to support the specification of data-oriented optimization processes. More specifically, in order to define each activity, the definition and notation given by the standard BPMN are followed. This implies that the activities participating in the optimization could be of any of the types described by the BPMN (script, manual, service, ...), and could also include any of the markers (sub-process, loop, parallel,...). On the other hand, the data objects defined in the formalization (ADI, ADO, PDI, and PDO) also follow the definition and notation given by BPMN. The DOOPT-DEC notation is incremented only in terms of the information carried by the arrows, which now include the name of the type of input and output data (ADI/PDI, and ADO/PDO). Therefore, DOOPT-DEC introduces the new data components related to the constraints (pre- and post-conditions, input and output constraints, and the objective function), whose definitions were previously limited through the grammar. All these components of the model are graphically represented by the symbols shown in Fig. 3.

2.4. Specification applied to the trip planner

In this section, the formalization is applied to the Travel Search subprocess presented in Section 1.1, in order to find the cheapest travel package that fits customer preferences.

In the example, there are nine PDIs whose values are provided by the customer as input data:

- **departingFrom:** city from where the customer departs.
- **setDepartingFrom:** set of possible departure cities where the flight can be taken.
- **goingTo:** destination city.
- **setGoingTo:** the set of possible arrival cities for the flight, containing the destination city itself and/or cities from where the destination city can be reached by using a rental car.
- **earlyDepartDate** and **lateDepartDate:** the earliest and last day when the customer prefers to depart, respectively.
- **earlyReturnDate** and **lateReturnDate:** the earliest and last day when the customer prefers to return, respectively.
- **hotelPreferences:** some preferences related to the hotel (single or double room, star-rated, etc.).

Given a data input (that is, specific values for the aforementioned eight PDIs), each activity calculates the price of its sought product. The activities and their ADI are detailed below.

- Flight Search Activity (A_F) returns the price of flight for a tuple of values for the data input. Fig. 4 shows the components related to the Flight Search Activity represented with DOOPT-DEC.
  DATA_INPUT (A_F) = {*departingFrom*, *goingTo*, *departDate*, *returnDate*}
  DATA_OUTPUT (A_F) = {*priceFlight*, *flightInformation*} where $flightInformation = \{outwardArrivalDate, returnArrivalDate, seat, ...\}$
  Certain existing pre and post-conditions include:

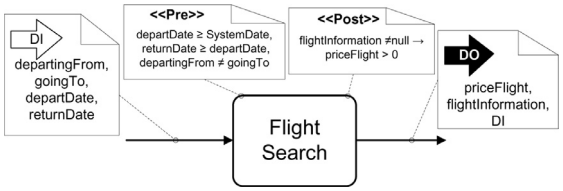

Fig. 4. *Flight Search* Activity specified with DOOPT-DEC.

$\mathbb{PRE}(A_F) = \{departDate \geq systemDate^1 \wedge returnDate \geq departDate \wedge departingFrom \neq goingTo\}$
$\mathbb{POST}(A_F) = \{flightInformation \neq null \rightarrow priceFlight > 0\}$

- Hotel Search Activity (A_H) is used to determine the cost of booking a hotel room. $\mathbb{DATA\_INPUT}(A_H) = \{location, checkInDate, checkOutDate, preferences\}$
  $\mathbb{DATA\_OUTPUT}(A_H) = \{priceHotel, hotelInformation\}$
  Certain existing pre and post-conditions include:
  $\mathbb{PRE}(A_H) = \{checkInDate \geq systemDate \wedge checkInDate < checkOutDate\}$
  $\mathbb{POST}(A_H) = \{hotelInformation \neq null \rightarrow priceHotel > 0\}$
- Car Rental Search Activities (A_{CR1} and A_{CR2}) are used to determine the price of renting a car. Two cars can be rented during the trip, one at the source (A_{CR1}), and another at the destination (A_{CR2}). Nevertheless, the price for renting both cars is represented by A_{CRx}, where x can take the values 1 or 2, and depends on these entries:
  $\mathbb{DATA\_INPUT}(A_{CRx}) = \{departingFrom, goingTo, departDate, returnDate\}$
  $\mathbb{DATA\_OUTPUT}(A_{CRx}) = \{priceCarR_x, carR_xInformation\}$
  Certain existing pre and post-conditions include:
  $\mathbb{PRE}(A_{CRx}) = \{departDate \geq systemDate \wedge departDate < returnDate\}$
  $\mathbb{POST}(A_{CRx}) = \{carR_xInformation \neq null \rightarrow priceCarR_x > 0\}$

In this case, the outputs of the process ($\mathbb{PDO}$) are the outputs of the activities, which contain the information about the elements composing the trip, as well as the total price of the trip (*price*).

Therefore, the customer only provides data to the subprocess: the set of dates/cities, and some preferences related to the hotel. Then, each search activity only uses the necessary data, as detailed in its specification, for the searching. On the one hand, the subprocess can have input data that is not used by some search activity: for example, the Flight Search Activity takes the possible dates and cities given by the customer to the subprocess, meanwhile, the Hotel Search Activity takes the possible dates, the destination city and the preferences. On the other hand, the subprocess can have input data that is not used in the same way by the activities: for example, the input data related to the departure date in the subprocess is a set of values, while is a single value in case of the Flight Search Activity. In both cases, the constraints between the input data of the subprocess and the activities are specified as $\mathbb{INPUT\_CONSTRAINTS}$ at design time. In the same way, the subprocess returns different data with respect to those returned by the activities, and it is necessary to define, at design time, how these data are calculated and related with the output data of the activities. Therefore, the $\mathbb{OUTPUT\_CONSTRAINTS}$ define these relationships between these output data of the activities and the data returned by the subprocess to the customer.

Although in the experimentation evaluation all the constraints have been included, only the most representative constraints have been formulated in this section. Regarding the relationships between the data input and output that belong to the subprocess and the activities, the most representatives $\mathbb{INPUT\_CONSTRAINTS}$ and $\mathbb{OUTPUT\_CONSTRAINTS}$ are detailed below:

- $\mathbb{INPUT\_CONSTRAINTS}$:
  - From the input data introduced by the customer, the possible values of the departure date ($IC1$) and return date ($IC2$) of the flights are obtained, satisfying the following constraints:
    ($IC1$) $earlyDepartDate \leq A_F.departDate \leq lateDepartDate$
    ($IC2$) $earlyReturnDate \leq A_F.returnDate \leq lateReturnDate$
  - The constraints that describe the possible values of the departure airport ($IC3$) and arrival airport ($IC4$) of the flight have to satisfy the possibilities of the input data proposed by the customer.
    ($IC3$) $A_F.departingFrom \in setDepartingFrom$
    ($IC4$) $A_F.goingTo \in setGoingTo$
  - The date of check-in into the hotel has to match with the arrival date of the outward flight ($IC5$).
    ($IC5$) $A_H.checkInDate = A_F.outwardArrivalDate$
  - The date of check-out has to match with the return date of the flight ($IC6$).
    ($IC6$) $A_H.checkOutDate = A_F.returnDate$
  - If the flight does not depart from the departure location ($IC7$), then the rental of a car (CR_1) is necessary.
    ($IC7$) $departingFrom \neq A_F.departingFrom \rightarrow \{A_{CR1}.departingFrom = departingFrom \wedge A_{CR1}.goingTo = A_F.departingFrom \wedge A_{CR1}.departDate = A_F.departDate \wedge A_{CR1}.returnDate = A_F.returnArrivalDate\}$
  - If the flight arrives at the destination city ($IC8$), then it is not necessary to rent a car.
    ($IC8$) $goingTo = A_F.goingTo \rightarrow A_F.goingTo = A_H.location$
  - If the flight does not arrive at the destination city ($IC9$), then the rental of a car (CR_2) is necessary.
    ($IC9$) $goingTo \neq A_F.goingTo \rightarrow \{A_{CR2}.departingFrom = A_F.goingTo \wedge A_{CR2}.goingTo = goingTo \wedge A_{CR2}.departDate = A_F.outwardArrivalDate \wedge A_{CR2}.returnDate = A_F.returnDate\}$
- $\mathbb{OUTPUT\_CONSTRAINTS}$:
  - The total price of the trip is the sum of all the prices returned by the activities, as presented in constraint ($OC1$).
    ($OC1$) $totalPrice = priceFlight + priceHotel + priceCarR_1 + priceCarR_2$

In this example, the optimization involves the minimization of the total price of the trip, which is composed of the cost of buying flight tickets, staying in a hotel room, and renting cars for the departure and arrival cities.

$\mathbb{OBJ\_FUNC} : (MIN,\ totalPrice)$

All these components of the Travel Search subprocess are represented in Fig. 5.

3. Data-oriented optimization model approaches

Previous sections present how to formalize the data optimization problem through the use of a special hybrid model. In addition, the aim of our proposal is also to reach a computable solution usable by the companies. It implies the possibility to deploy the hybrid model in a commercial BPMS, therefore it should be totally described using an imperative model. To build this imperative model, it is necessary to analyze the relation between the $\mathbb{PDI}$, $\mathbb{PDO}$, $\mathbb{DI}$, $\mathbb{DO}$, all the constraints and the optimization function.

One of the aspects to analyze in deep is how the possible values for the $\mathbb{ADO}$ and $\mathbb{PDO}$ can be obtained for each possible $\mathbb{ADI}$ and $\mathbb{PDI}$ combination (with or without executing the activities). For the Travel Search subprocess, the objective function is to minimize the *totalPrice*, which, a priori, can only be known when the activities are executed at runtime and the values for the *priceFlight, priceHotel, priceCarR*$_1$, and *priceCarR*$_2$ are calculated. However, it would be also possible to know the output values of the activities using the input values and the pre and post-conditions. For example, if the

[1] The variable *systemDate* corresponds to the date and time zone in which the customer is located when requesting the search.

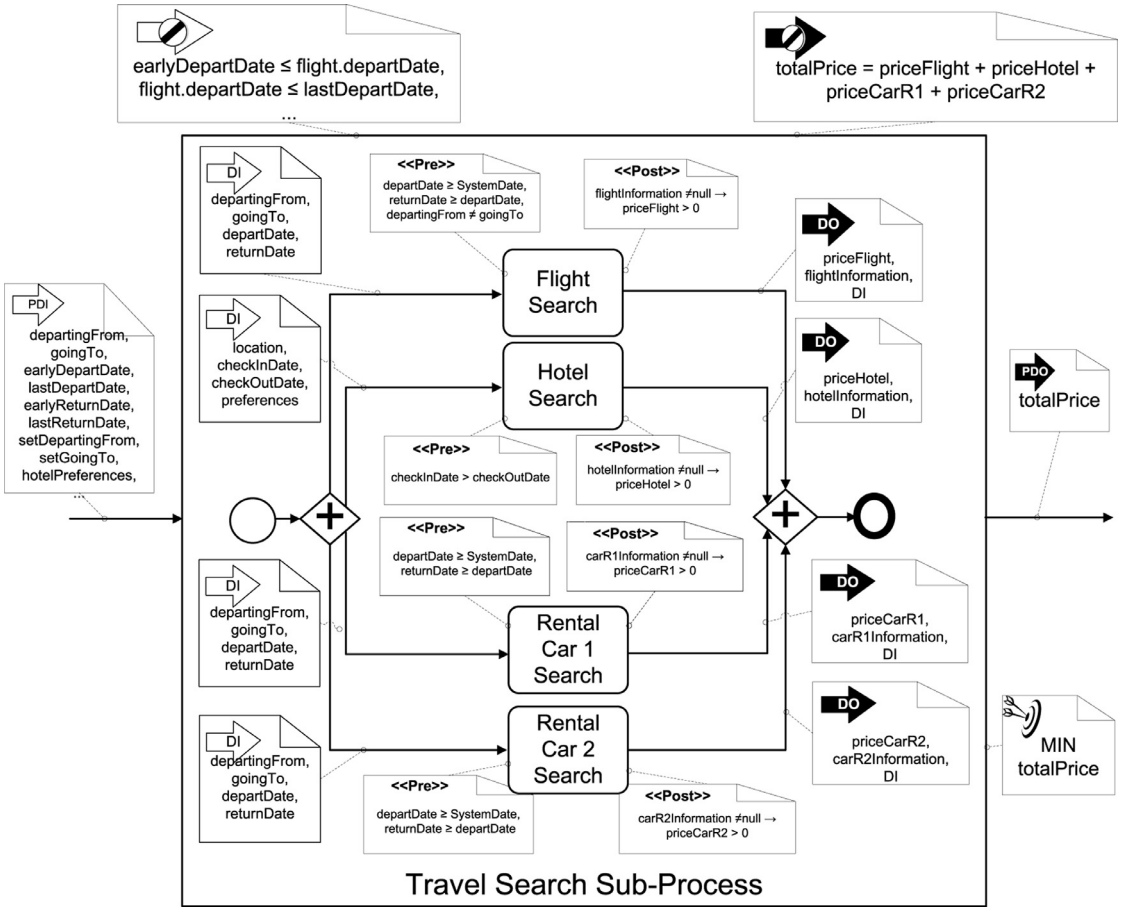

Fig. 5. Example of trip planner described using DOOPT-DEC.

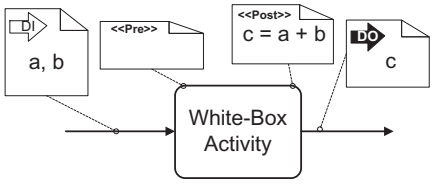

Fig. 6. White-box Model Example.

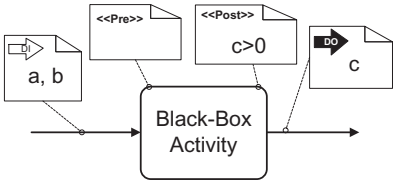

Fig. 7. Black-box Model Example.

priceHotel can be calculated by means of a function described in a post-condition (i.e. $priceHotel = (checkOutDate - checkInDate) * 50$), and the same occurs with the rest of the prices. In these cases, it is unnecessary to execute the activities to calculate the *totalPrice*. This implies the possibility of finding the best input data to optimize the outcome including every constraints in a COP (see Section 3.1), and solving it in a local manner. By avoiding the execution of the activities, the complexity and the way in which the problem is solved in an imperative model is reduced and simplified. If it is not possible, every possible $\mathbb{ADI}$ combination must be generated and used as input data of the activities to obtain the outputs. Therefore, regarding the knowledge about the behavior of the activities described by the pre and post-conditions, two different imperative models could be created:

- *White-box model*: This is used when the subprocess is formed by *white-box activities*, which entails that the output values of the variables can be obtained in terms of the input values and the pre- and post-conditions, and it is unnecessary to execute an activity to ascertain its output, as shown in Fig. 6. In this case, the optimization problem can be modeled and solved in a local manner, since the execution of the activities is unnecessary.
- *Black-box model*: This is used when the problem is formed by *black-box activities*, which entails that, given a set of input values, the pre and post-conditions of the activities are insufficient to ascertain the output values, as shown in Fig. 7. In this case, it is necessary to execute the activities to obtain the specific values for the data output, since the model is unknown by the BP itself until the execution. However, the creation of all the possible combinations of values for the data input can be modeled and solved locally. Using these combinations of input data values in the execution of the activities, it is possible to obtain the output data used in the objective function that optimize the outcome of the process.

The black-box approach is faced in [50], where an executable architecture is presented to solve the combination and execution of activities for the optimization of the objective function. Therefore, in this paper, we focus on solving only the white-box approach.

3.1. Background in constraint satisfaction problem

A Constraint Satisfaction Problem (CSP) represents a reasoning framework consisting of variables, domains, and constraints. Formally, it is defined as a tuple $\langle X, D, C\rangle$, 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 $\{d(x_{k_1}), d(x_{k_2}), \ldots, d(x_{k_l})\}$ an *instantiation* of the variables in V_k, then an instantiation is a solution if and only if, it satisfies all the constraints C. If an objective function O, is included in a CSP, it is called a Constraint Optimization Problem (COP), defined as a tuple $\langle X, D, C, O\rangle$. A COP is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables. The objective function implies maximizing or minimizing a variable that can represent a numerical combination of others by means of a function. The details of how the COP is created are explained below.

In order to solve a COP, a combination of search and consistency techniques is commonly used [10]. 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 analyzes the combination of values of variables where the constraints are satisfiable and where the values that cannot improve the best solution found until the moment are eliminated. Several local consistency and optimization techniques have been proposed as ways of improving the efficiency of search algorithms [71].

3.2. How to obtain an imperative business process model from a white-box specification

In order to create and solve an entirely imperative model using the definitions in Section 2, this contribution proposes to introduce in the model a COP with the constraints, data, and the objective function. The focus is placed on how a COP can be automatically built. Although the creation of the COP is performed only once at design time, the resolution of the COP occurs for each instance at runtime, to obtain the optimized outcome in function of the customer requirements.

The use of Constraint Programming enables the combination of the variables to represent the model and ascertain those values of the output data that optimize the objective function. We propose that this COP can be set-up as a script activity (as shown in Fig. 8) within an executable model to manage the possible values of the input data variables of the activities.

Firstly, it is necessary to highlight that the specification of the COP is very similar to the formalized DOOPT-DEC model, since both are declarative models that define the problem but not how to solve it. The advantage of using a COP is that it is not only a way to represent the problem, but it also provides a way to find a solution through a constraint solver (Choco Solver [7] in our case), and, at execution time, to find the concrete values that optimize the defined function. In order to design a model that involves all the variables and constraints that describe the data-oriented optimization process, the following COP structure is proposed:

- X: $\mathbb{DF}$, which includes the variables corresponding to each item of the data belonging to $\mathbb{PDI}$, $\mathbb{PDO}$, $\mathbb{DATA\_INPUT}(A_1), \ldots, \mathbb{DATA\_INPUT}(A_n), \mathbb{DATA\_OUTPUT}(A_1), \ldots, \mathbb{DATA\_OUTPUT}(A_n)$.
- D: Domain depending on the $\mathbb{DF}$ variables. The values of the domain depend on each instance of the BP.
- C: $\mathbb{PRE}(A_1) \wedge \mathbb{POST}(A_1) \wedge \ldots \wedge \mathbb{PRE}(A_n) \wedge \mathbb{POST}(A_n) \wedge \mathbb{INPUT\_CONSTRAINTS} \wedge \mathbb{OUTPUT\_CONSTRAINTS} \wedge$ {Instantiation of the $\mathbb{PDI}$ with the values of the specific instance}
- O: $\mathbb{OBJECTIVE\_FUNCTION}$

The automatic creation of the COP from the *subprocess description* and *subprocess relationships* is carried out using a model-to-text transformation. In our case, Epsilon [16] has been used. Among other things, Epsilon provides a family of languages and tools for code generation and model-to-model transformation. The automatic creation consists of going over each element of the DOOPT-DEC language and transforming it into the corresponding text in the COP. With the aim of being independent of the technology used, DOOPT-DEC language is proposed, and then, to allow its execution, this language should be translated into the specific language used by any of the existing CSP solver. This translation is always straightforward and easy. Firstly, each element of the DOOPT-DEC language has been detailed in this paper, and the specific elements of the COP depends on the CSP solver chosen. Secondly, Epsilon provides a language, such as ETL [17], which allows us to define the correspondence between the elements of both languages as explained before. And finally, once a specific instance is defined in the DOOPT-DEC language, the automatic creation is carried out and the resulting COP is obtained.

As mentioned previously, the created COP is included as a *script activity* [48] within an imperative model in order to evaluate the declarative data information at instantiation time. This COP provides the best input data for the activities that optimizes the outcome data of the subprocess. The basis of the automatic creation is to use the *subprocess description* and *relationships* in order to create a COP in a script activity connected to the subprocess. Then, this imperative model can be executed in any of the commercial BPMSs. Therefore, the configuration of this script activity depends on the mechanisms provided by the modeling tool of the BPMS used, and also on the BPMS engine capabilities.

Regardless of the BPMS used, the COP can be solved using any of the existing COP solvers which connects the script activity with an external solver.

Finally, the resulting business process, for any customer requirements, is capable of searching for the cheapest travel plan for a given set of dates and cities. In other words, the result after having solved the COP in our running example is the combination of dates and cities in which the customer should travel. This combination is the one with the minimal price, which meets the objective function.

3.3. Implementation of white-box model approach

In our case, the BPMS employed to design and execute the BPs is Bonita Open Solution^{TM} [8] since it is an open-source application with a free distribution, and is commonly used in the private market. We have created a *connector*, which is the way in which Bonita Open Solution^{TM} links an activity with a service or application that executes a functionality. The new connector, called *COP Script*, enables the inclusion of a COP code into an activity and permits its resolution. *COP Script* has been implemented with the libraries provided by the COP solvers, Choco Solver [7] in our case. The business modeler can therefore use this connector to link the COP solver with the script activity. Fig. 9 shows an example of how a business modeler can include the generated COP into any script activity using the *COP Script connector* in Bonita Open Solution^{TM}.

Finally, at runtime, when an instance reaches the script activity with the specific valuation of data, the COP is solved and the result is obtained[2].

[2] The code of the resulting COP is available at http://www.lsi.us.es/quivir/index.php/Main/WhiteBox. This COP must be included as a script task in a Bonita Open Solution^{TM} project using the COP script connector, and the Choco solver Choco web-site.

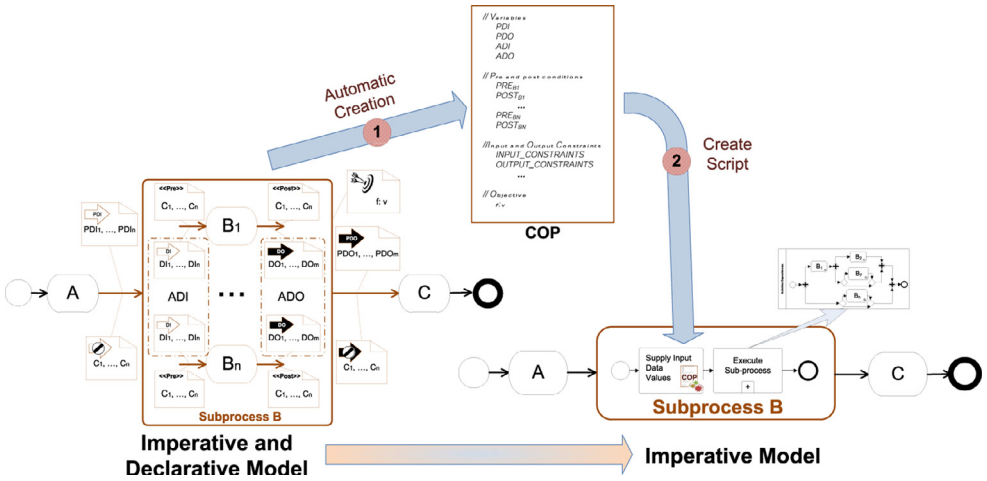

Fig. 8. White box transformation.

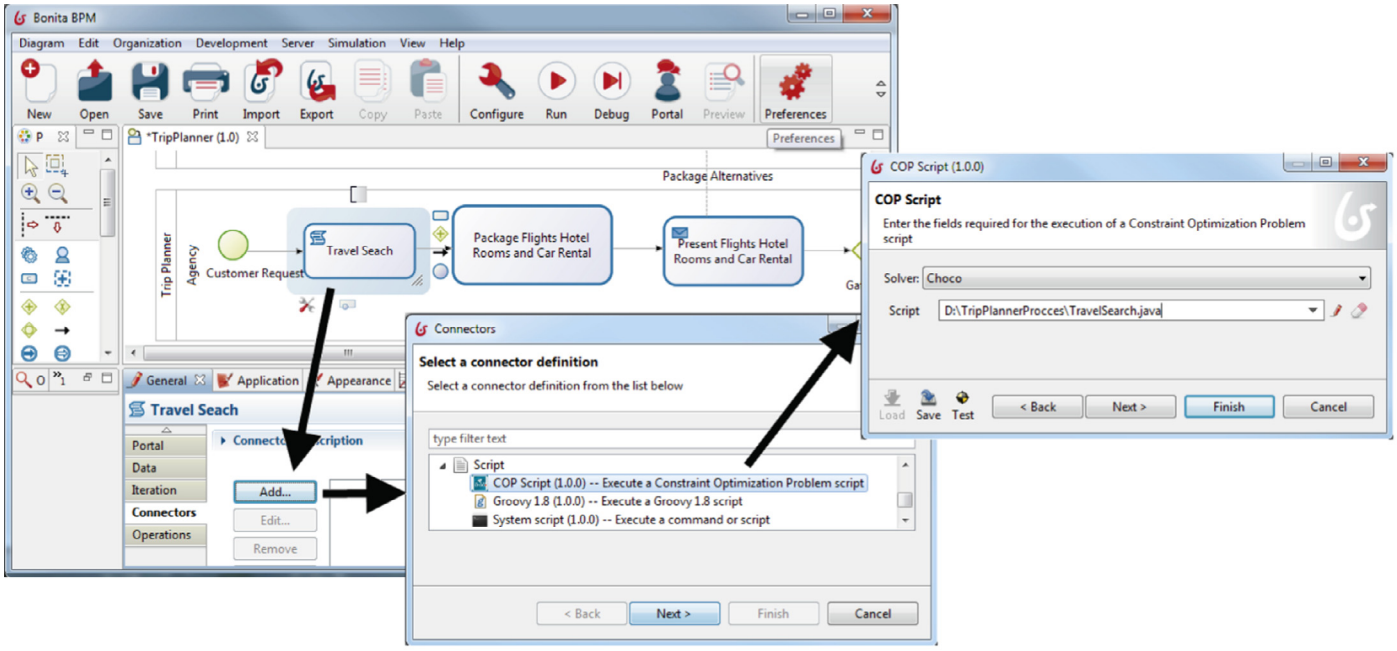

Fig. 9. Choco Script Task Configuration in Bonita Open Solution™.

3.4. White-box model approach applied to trip planner

A clear example of white-box approach in the trip planner example occurs when the cost of each part of the travel is included in different travel brochures, such as the offers available at travel agencies. In this case, since the prices are pre-established for specific date and conditions, the sub-process knows, without executing the activities, the data input and output relation for each part of the trip. For example, all flights from Seville to London, regardless of the date, costs 100 euros in May ($A_F.departDate \geq$ "$2014-05-01''$ $\wedge$ $A_F.departDate \leq$ "$2014-05-31''$ $\longrightarrow$ $priceFlight = 100$). Since the travel brochures are accessible and public, and cannot change for the dates published, then this information can be included in the COP. The COP generated from the model specified in Section 2.4 is shown in Table 1.

4. Empirical evaluation

In this section, the experimental results corresponding to the performed evaluation are shown, the obtained results are discussed, and the limitations of our proposal are laid out.

4.1. Experimental design

Since the generated COP is created depending on the declarative model, it is formatted only once, and at design time. The COP creation is performed linearly in accordance with the size of the model, that is, the number of variables and constraints. For this reason, it is not time-consuming. The execution problem, and the bottleneck involved, is the resolution of the COP at runtime, where the critical point of our proposal lies.

Since the optimization of the output has been mapped into a COP, the time spent during the optimization is linked to the complexity of solving this COP. This has been analyzed in great depth over recent decades [6], and depends on two parameters: the width of the graph and the order parameter. On the one hand, the width of the graph represents the relation among the constraints, where the tractability in CSPs is due to the structure of the constraint network, and where the tree-structured CSPs have polynomial complexity (linear with respect to the number of variables, and quadratic with respect to the cardinal of the domain of the variables). On the other hand, the order parameter, defined as the ratio of the number of forbidden tuples to

Test	Set Departing From	Set Going To	Set Departure Date	Set Return Date	Number of Combinations	Number of solutions
Test00	1	1	1	1	1	1
Test01-10	[2-30]	1	1	1	[2-30]	[2-30]
Test11-20	1	[2-30]	1	1	[2-30]	[2-30]
Test21-30	1	1	[2-30]	1	[2-30]	[2-30]
Test31-40	1	1	1	[2-30]	[2-30]	[2-30]
Test41-50	[2-30]	[2-30]	1	1	[4-900]	[4-846]
Test51-60	1	1	[2-30]	[2-30]	[4-900]	[3-621]
Test61-70	[2-30]	1	[2-30]	1	[4-900]	[4-764]
Test71-80	1	[2-30]	1	[2-30]	[4-900]	[4-830]
Test81-90	[2-30]	1	1	[2-30]	[4-900]	[4-830]
Test91-100	[2-30]	[2-30]	[2-30]	1	[12-27900]	[12-21601]
Test101-110	[2-30]	[2-30]	1	[2-30]	[12-27900]	[12-27900]
Test111-120	[2-30]	1	[2-30]	[2-30]	[12-27900]	[12-27621]
Test121-130	1	[2-30]	[2-30]	[2-30]	[12-27900]	[12-27897]
Test131-150	[2-30]	[2-30]	[2-30]	[2-30]	[18-1102500]	[18-1043111]

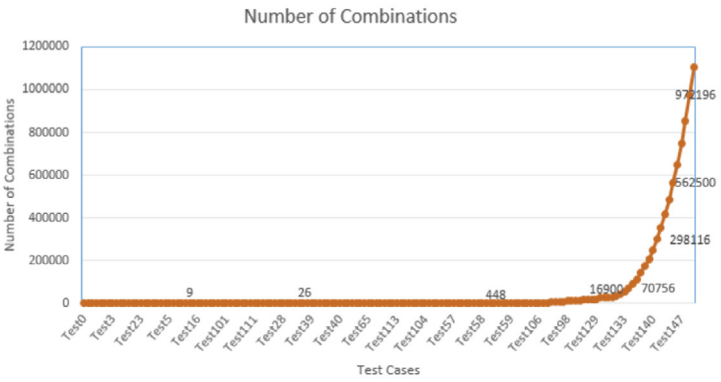

Fig. 10. Number of possible combinations for each test case.

Table 1
Constraint optimization problem for outcome data optimization.

```
//Variables:
  earlyDepartDate: Date
  lastDepartDate: Date
  priceFlight: Integer
  ...
//Pre and post-conditions
  A_F: A_F.departDate > SystemDate
    A_F.departDate ≥ "2014 − 05 − 01" ∧
    A_F.returnDate ≤ "2014 − 05 − 31" → priceFlight = 100
  ...
  A_H: A_H.checkOutDate > A_H.checkInDate
    A_H.checkOutDate > A_H.checkInDate
  ...
// Input and Output Constraints
  A_F.returnDate = A_H.checkOutDate
  earlyDepartDate ≤ A_F.departDate
  A_F.departDate ≤ lastDepartDate
  ...
//Optimization function:
  TotalPrice = priceFlight + priceHotel + priceCarR1 + priceCarR2
//Objective
  minimize(totalPrice)
```

the total number of possible combinations, determines the partition of the problems space into under-constrained (several possible solutions), over-constrained (no possible solutions) and just-constrained (a set of a low number of possible solutions) problems. In the first two cases, the problems are scalable, but in just-constrained problems, a significant increase of solving cost could occur and the scalability would not be possible [29]. For these reasons, no general affirmation can be given concerning the efficiency in a generic way of our proposal, since our declarative specification permits any type and any number of numerical constraints; therefore, the evaluation time depends on the specific problem. Depending on the number of constraints associated to a COP, and the number of variables, the COP evaluation time remains variable.

For this reason, and with the aim of performing the evaluation of our proposal, the resulting COP of the case study is executed over a set of 150 generated test cases. Each test case establishes a different number of values for the $\mathbb{PDI}$, since the variables provided to the process determine the complexity of the resolution and the number of possible combinations. Likewise, the different values for each data in $\mathbb{PDI}$ generate, in turn, a number of combinations for $\mathbb{PDI}$ variables. This implies that each test case considerably increases the number of combinations, as shown in the table and plot in Fig. 10. Since some of the variables of the $\mathbb{PDI}$ are sets, depending on the different possibilities, the number of combinations increases whenever the sets grow. For example, if the "*earlyDepartDate*" is "23-12-2014" and "*lateDepartDate*" is "13-01-2015", then there are 22 possible days to depart, and this implies different combinations with the rest of the variables, such as *returnDate*, *departingFrom* and *goingTo*. Since not all the combinations are possible solutions of the problem, Fig. 10 also indicates the number of possible solutions for each test. For example, it is possible that there is no available flight to go to any of the destination airports for a specific combination of dates. In that case, this combination is not a solution to the problem and should be discarded.

4.2. Experimental result

The main purpose of the experimental evaluation is the determination of the computing time (seconds), the memory used (MB), and the quotients between the time and used memory needed to solve the COP of the example. In order to highlight the importance of the optimization, the measurements have been carried out over the COP. The study has been made with a branch and bound search (called *optimized* option) and without it (called *all solution* option) to analyze how the consistency techniques used in COPs can improve the evaluation time.

The test cases were measured using a PC with an Intel Core i7-2675QM CPU with a 2.2 GHz processor and 8.00 GB of RAM.

On the one hand, Fig. 11 shows the memory used to solve the COP. It is possible to notice that, for the *optimized* option, the memory used almost remains steady regardless of the number of permutations. Meanwhile, the behavior of *all solution* option is exponential.

On the other hand, the behavior of the execution time also fits an exponential model for *all solution* options (see Fig. 12). However, it is necessary to highlight that the evaluation time almost remains steady for the various tests, although the number of possible combinations for the $\mathbb{PDI}$ variables increases exponentially for the different tests presented. It is possible thanks to the use of propagation and consistence techniques in Constraint Programming.

4.3. Scope of applicability of our proposal

In this paper, the focus is on data-oriented optimization processes, whose main goal is to obtain an outcome data optimization starting from the specification of the data and the constraints involved in the process. As a consequence of the declarative model described in the previous subsections, the scope of application of our proposal is limited by various characteristics:

- *The knowledge concerning the possible values of the variables managed during the business process instance*. If the relation of the values of the input and output variables are unknown, then the resolution of the COP is insufficient to help the customer during the business execution to achieve an objective. Since the inference of

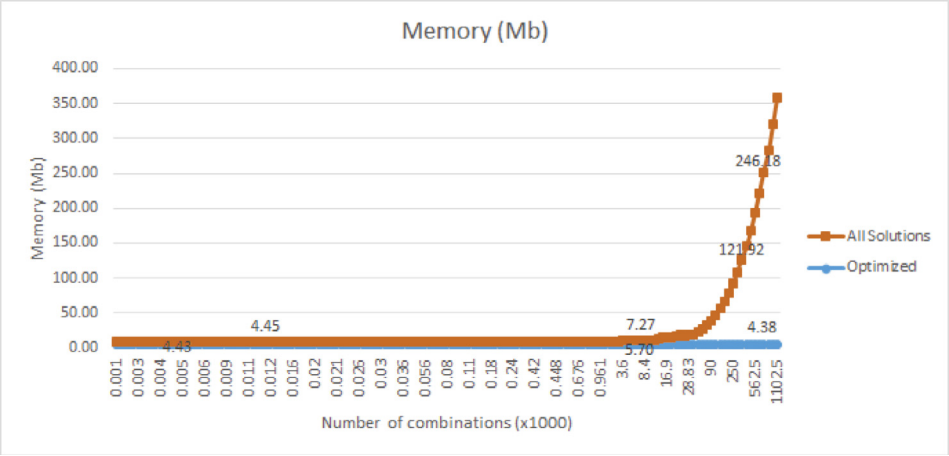

Fig. 11. Memory used for the tests of the example.

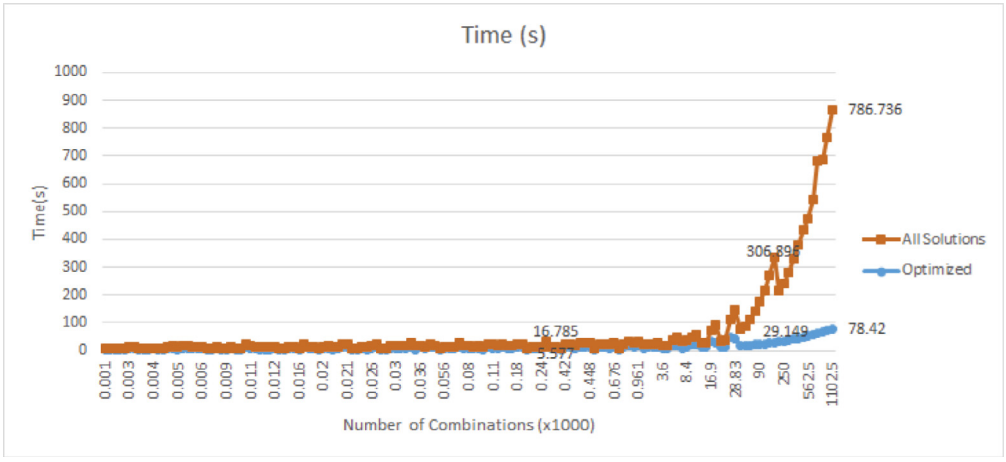

Fig. 12. Time used for the tests of the example.

the possibility of correct input values is derived from the constraints associated to the model, if these constraints remain unknown, then the execution of the activities is mandatory and our solution of a black-box framework [50] will be necessary.

- *The possible constraints that can be defined, since the limitation is centered on the capacity of their expressiveness as allowed by the grammar and the type of variables*. These constraints constitute the formal representation of the relations between the data that flows through the BP. The limitations of use of the proposal appear when the constraints cannot be represented by the relations described; by means of the presented grammar; by the data type; or by the operators included in this proposal, such as when a relation between two variables is described by means of a trigonometric function. The limitation of the data domain and the operations that can be applied is established by the solver for the COP, explained in Section 3.1. Thanks to the variety of types of variables, the existing commercial constraint solvers are capable of solving real problems using: Float, Integer, Sets, Boolean, Date and String variables.
- *Knowledge of the order in which the activities are executed*. The uncertainty related to the imperative model comes from the necessity to specify the relationship between activities through the data handled, since the optimization function depends on these data and relationships. The activity order is irrelevant since the optimization is centralized in the resolution of the COP. In the presented proposal, the activities are executed in parallel since their input data related to dates and cities are single values and the optimization problem can be centralized in the script activity. However, if the input data of the activities related to dates and cities is a set of possible values, then each activity can establish the input data that optimize the output with different combinations of values and the order between the activities becomes relevant. In that case, the order between the activities is unknown because it depends on the values of the data handled in each case, and the optimization process is more complex. The complexity is based on the need of analyzing the process for each instance and if the model has to change because it does not support this instance, then the modeling, development, deployment, and execution of a new model in a BPMS is necessary. All these steps cannot be done at runtime since it requires design modifications, which involve highly time-consuming.

5. Related work

There are many languages that enable the description of BPs in a declarative way. In this section, the most important declarative languages are analyzed, and various approaches, related to the transformation between imperative and declarative models, have also been included.

5.1. Comparison of declarative languages

BPs are specified by using modeling languages, commonly with graphical notations. The selection of an adequate graphical method

has become a major issue for both academic researchers and business professionals, since each individual process modeling method has its own characteristics. As a consequence, there are many research efforts dedicated to improve modeling methods. In [25], a comparison of most of these graphical process modeling methods is presented.

Many languages enable the description of a BP in an imperative way. The main differences between them are the elements that can be used, and the information that can be included in the model. One of the most important assets of imperative languages is the ability to be executed in a BP since the imperative model has to be enriched to be executed, for example, by adding the connectors to the implemented activities or the details of connection for the database accesses.

On the other hand, although processes are typically specified in an imperative way, declarative process languages have been increasingly used. Declarative descriptions are easier to use for some examples (such as medical guidelines and policy compliance), since they enable the specification of what has to be done instead of the specification of how it has to be done. However, imperative models remain easier to understand since they explicitly enumerate the allowed execution traces, as it is maintained in Weber et al. in [72] and Zugal et al. in [77]. Several studies have been made to know how humans can understand the two types of BP models.

In [52], Pesic performs an in-depth analysis into workflow flexibility, where flexibility is related to the capacity to support changes at runtime. The flexibility of our proposal is based on possibility of modifying the optimization function and the values for the input data at runtime without having to change the imperative model for each case. Imperative models lack expressiveness, flexibility, and adaptability, since they enumerate the allowed execution traces at design time. When the specific control-flow or data-flow of a business process cannot be described at design time, declarative languages are used. They describe a process in terms of a set of constraints that must be satisfied during the process execution. Most of the declarative models describe that every activity can be freely performed unless it is forbidden.

5.1.1. Main characteristics of declarative languages

Certain characteristics, considered relevant in the existing declarative languages, have been analyzed in order to ascertain whether they enable data-oriented optimization problems in BPs. In order to sort and group the various proposals found about declarative languages, we have considered the following characteristics:

- *Formalism for reasoning*: The proposals use different formalisms for reasoning. Although most of them use one type of formalism, sometimes they combine more than one, and/or are improved by means of made-to-measure[3] algorithms. The most relevant formalisms for reasoning used in the declarative languages include: (i) *Linear Temporal Logic* (LTL), which represents desirable or undesirable patterns within a history of events [51]; (ii) *Event calculus*, which is a logic programming formalism to represent and reason on the effects of events on the state of a system [28]. This formalism also has the ability to reason abductively and therefore produces a sequence of transitions that must happen for a particular instance to hold in the future [47], and; (iii) *Constraint programming (CP)* which permits the description of the model by means of the variables and the constraints that relate these variables (see Section 3.1).
- *Hybrid description by using Imperative and declarative combination*: Some languages have the capacity to describe aspects in the same model, both imperatively and declaratively. When the order of the activities can be described an imperative description is used, but when we only count on a set of business rules that describe which activity relations are permitted, and which are prohibited, then a declarative description is used.
- *Use of the model*: The engines that support the declarative proposals permit various actions, such as: *Validation* of the model for a trace of events, *Construction* of automatons to generate a possible sequence of activities, and *Assistance* to the customer to decide which activity is the best to execute at runtime.
- *Data perspective*: Some of the languages enable the values of the data-flow variables to be included in the rules that describe the declarative model, to determine the activities order when it depends on the data values at runtime.
- *Pre and post-condition*: The inclusion of a description of how the data are modified during the BP execution, by means of pre and post-conditions, is a very relevant capacity. This allows the modeler to describe the possible data values before and after each activity or subprocess execution. This information tends to be used for execution validation.
- *Optimization function*: The possibility of including an optimization function in the description that is taken into account in the model.

5.1.2. Main declarative languages

The most important declarative languages are summarized below.

- *Pocket of flexibility*. This solution is based on constraint specification of the business process workflow [63]. In the framework, called *pocket of flexibility*, it is possible to combine activities whose relation is known with activities whose relation remains unknown. This relation is done by means of a *build activity*, which provides the set of constraints for the specification of the pocket with a valid composition of workflow fragments. Although data is included in the activity-level constraints [64], these constraints no longer permit the activity functionality to be described nor the objective function to be optimized. A report in 2013 [76] shows that this research line, related to workflow technology, has been abandoned by the Sadiq et al. research group.
- *DeCo (Declarative Configurable Process Modeling Notation)*. Irina Rychkova et al. in [59–62] present a declarative BP specification language that enables designers to describe the actions that a business process needs to contain. For every action of the working object, the specifications define a pre-condition and a post-condition. The pre and post-conditions represent how the different actions can modify the state of the objects transformed during the process execution; they do not define the order of the actions. Furthermore, these proposals fail to include the specification of a business goal as the cornerstone of the modeling and its subsequent execution.
- *Compliance Rule Graphs (CRGs)*. The CRGs [27,31,33] focus on finding an appropriate balance between expressiveness, formal foundation, and efficient analysis. For these reasons, the authors propose a language based on a graph representation where the order of the activities and the occurrence or absence of activities can also be included. Thanks to the specification of the compliance rules of the BP, the set of all possible execution traces of processes based on activities are defined [55]. Awad et al., in [2], extend the capacities of CRGs by including data in the specification of compliance rules, which the authors called data rules. However, this data perspective fails to enable the model to be focused on the search for the optimal business goal.
- **Em-Bra^{2}Ce.** The Em-Bra^{2}Ce Framework [20,57] presents a declarative language based on the standard SBVR (Semantics of Business Vocabulary and business Rules) [65]. The basis of the Em-Bra^{2}Ce Framework is the specification of the state space and the set of business rules [20–22]. Thanks to the business rules, not only can the sequence of activities and the rights and duties of agents and roles be determined, but the events can also be

[3] Algorithms developed only for a single proposal: to solve a specific problem and develop a specific algorithm for each model, being necessary the participation of an expert to design it in each case

Table 2
Comparison of declarative languages .

	Formalism	Imper. and Decl.	Use of model	Data persp.	Pre and Post	Opt. Funct.
Sadiq	Graph theory	✓	Val.			
DeCo	First Order Logic	✓	Val.	✓	✓	
CRGs	Pattern matching		Val.	✓		
Em-Bra^{2}Ce	Colored Petri Net	✓	Val.	✓		
Penelope	Event calculus	✓	Constr.			
ConDec	LTL		Val.			
ConDec-R	Constraint Programming		Assist.	✓		✓
Data-aware	Event Calculus		Val.	✓		
Case Handling	-	✓	Constr.	✓	✓	
DOOPT-DEC	Constraint Programming	✓	Constr. Assist.	✓	✓	✓

controlled, the business concepts are defined and constrained, and the policies and regulations can be motivated. However, the Em-Bra^{2}Ce Framework fails to enable the BP to focus on the objective to be optimized.

- *Penelope*. Penelope [18] expresses temporal rules about the obligations and permissions in a business interaction using Deontic logic. Penelope enables sequencing and timing constraints to be defined between the activities. Therefore, the only type of data that is included in the definition is related to the execution time of the activities, since the data managed during each instance is not an object of the proposal. This language is supported by an algorithm to generate compliant sequence-flow-based process models that can be used in business process design [18,19].
- *Declare*. The Declare language (previously also presented as ConDec) was designed for modeling and enacting dynamic business processes [51,53]. Declare is a constraint-based language, since it is focused on defining the activities involved in the process and the execution constraints between the activities, by means of order relations between these activities. Declare has been used to validate a sequence of events [75], runtime validation [34,35], monitoring [45], runtime verification [46], validation and monitoring at runtime [53], and guidance of users through recommendations [66].
- *ConDec-R*. ConDec-R is an extension of the Declare language for the inclusion of a description of the resources necessary during process execution [3]. The implementation extension assists the customer by means of recommendations to achieve an optimized plan for one or multiple objectives [26]. In order to obtain the plan, a Constraint Programming paradigm is used, combined with a set of algorithms to minimize evaluation time. Although this proposal incorporates the resource perspective, which is a type of data, this type of information is not oriented towards activity input and output data.
- *Data-aware constraints in declare*. This is another extension of the Declare framework [44] that permits the representation of the input, internal, and output data of the activities in a declarative representation of the order of the activity. Although the data aspect is included, the pre and post-conditions and the objective function are all excluded from this description. Montali et al. also propose a technique to automatically determine declarative process models that incorporate both control-flow dependencies and data conditions. However, these dependencies and data conditions are not focused on the optimization of the outcome data, the authors looked for a ready built declarative model to represent all the possible sequence of activities depending on the data values.
- *Case handling*. The Case handling paradigm [69] focuses on what can be done to achieve a business goal depending on the possible scenarios. Case handling focuses on a single process instance, called a *case*, by describing and specifying the ordering of activities depending on this case. To handle a case, the set of activities needs to be executed, and case handling involves humans to perform atomic tasks but has an orchestration engine to perform the orchestration (coordination) work. The data objects can remain undefined, defined, or unconfirmed, which affects the behavior of the activities. Graphical languages, such as Petri nets [70] and workflow graphs [74], are used to define this order between the activities [68].

Although all these declarative languages include information about data, none includes the data input and output of the activities for the optimization of the outcome of the business process. As shown in Table 2, none of the declarative languages includes all the characteristics presented previously in its model. Each declarative language uses a different formalism of reasoning, since each enables a control-flow sequence to be attained based on the declarative components specified in the model to solve the data-optimization problems proposed in this paper. Half of the proposals enable hybrid definitions with imperative and declarative components in the same models. Most proposals use the declarative models to validate whether a sequence of activities, for a given instance, is correct based on the BP requirements. Although the objective function is considered by ConDec-R, it is focused on the optimization of time and resources, and not on the outcome data of the BP to be offered to customers.

Furthermore, it is necessary to highlight that these declarative languages are not concerning about supporting the execution of any instance at runtime as occurs in imperative models. Declarative languages only provide tools to guide and recommend how a specific instance must be executed. All these characteristics render DOOPT-DEC as a complete alternative for the specification of declarative requirements within imperative descriptions for data-oriented optimization problems in BP.

5.2. From declarative to imperative descriptions

Since an automatic creation from a declarative specification into an imperative specification (more specifically, into a script task) has been carried out, it is interesting to analyze the studies that make some kind of transformation between models in business processes arena. In general, papers found in the literature related to the transformation from declarative to imperative descriptions are not focused on the assistance of the customer for the input data. Only in [24] the decision-making support is oriented towards data input, but it omits optimization of an objective function, and only makes the BP instances satisfiable for imperative models. In other works related to how to model the processes, such as [3], Barba et al. propose a framework of assistance to create models which take the necessary resources involved in the process into account. In that contribution, the data that describes the resources of the execution of the process are used, but not the data that flows at runtime, nor does it consider the input data for the activities that constitute the process. On the other hand, there are various studies that apply transformations to BP modeling. Victoria Torres et al. in [67] apply transformations to generate the navigation between web pages from a BP definition, specifically the BPEL executable description that implements the entire process. In [13], Fabra et al. integrate a service-oriented development method

(SOD-M) and a platform for the development and execution of interoperable dynamic web processes (DENEB). They present a model-driven framework for the analysis, design, development and execution of BPs, which covers BPM solutions from the very early stages of their development to their deployment and execution. Although a certain number of these papers are focused on defining the interaction between various participants in order to achieve business goals, none of them deals with the type of problems presented in this work: the creation of a data-oriented optimization imperative model from a declarative model.

The necessity to combine declarative and imperative perspectives in a hybrid model was analyzed in previous work, such as in [56] by Reijers et al. That paper reports the opportunities for practitioners of using a hybrid model instead of a completely declarative or imperative model. These researches carried out an experiment with ten professionals from industry with the aim of highlighting the strengths and weaknesses of the two approaches independently, and the potential of hybrid model that includes both perspectives. They also proposed a research agenda for the application of hybrid techniques. Maggi et al. in [38] subsequently presented a technique for the determination of a hybrid process model from an event log. In that paper, the declarative language used was Declare [40], and the imperative language was Pocket of flexibility [63]. Other work, such as [39] and [30] used hybrid techniques for the verification of business processes but not for the specification of the model itself. In [11], De Giacomo et al. describe a mixture of declarative and imperative process modeling styles, but only focused on the activities to execute. The BPMN-D language proposed permits to combine, in the same model, imperative and declarative descriptions related to the order between the activities. Therefore, the hybridization is not related to data values. On the other hand, De Masellis et al. in [12] enrich the language Declare with data-aware constraints. In that paper, the data is used to verify the temporal evolution of data according, but not to find the most optimized value of the data for each instance.

All these studies use hybridization based on activities, but to the best of our knowledge, there are no proposals that address the data optimization problem in hybrid models as we have described in this article, where the declarative part is oriented towards the data relationship specification, and the imperative part is focused on the control-flow perspective.

5.3. Role of data in execution time

One of the bases of our proposal is the importance of data instantiation at runtime, since the values are ascertained at runtime from a given specific set of input values. Therefore, our proposal could be seen as a recommendation system since the most appropriate values for the data input belonging to an activity are found automatically. In the literature, it is possible to find several recommendation systems which enable customers to be guided towards executing an instance with the aim of obtaining the business goal. Conforti et al. in [9] establish a recommendation system in order to predict possible risks in the model. The event log is analyzed in order to study the sequence of activities executed and the specific values consumed and provided by these activities. This analysis establishes the occurrence of possible faults (excessive duration, reputation-loss, and cost overrun), and provides a decision support for risk reduction. However, the proposal needs historical information extracted from the event logs in order to provide a solution. In our case, historical information is invalid since the output of the activities depends on the constantly changing input values and company policies, and historical information is only reflected in the post-conditions. At the same time, Maggi et al. in [37] also establish a predictive system in order to prevent customers from executing an instance that would fail to obtain the desired business goal. The authors establish a system to recommend which activity must be performed and what data input values the customer must provide. The main discrepancy is that, in their case, the business goal is to help in decision-making (whether to perform a surgery or not), but in our case, the business goal is to obtain the optimal outcome, which is a function depending on the data values. On the other hand, Castellanos et al. in [5] proposed automatic enrichment of business processes by including time metrics obtained from predictive methods. These time metrics enable both planning and decision making to better meet the goals of the business process. The optimization proposed by the authors is focused on setting the model parameters and choosing the best of these parameters. However, these parameters are only oriented towards aspects of time. This kind of optimization implies a probabilistic study which includes a confidence value, a prediction (acceptable or unacceptable), and a certain threshold in order to make plans. Therefore, the main obstacle for our kind of process is that, in data-oriented optimization problems, the probabilistic study cannot be included.

6. Conclusions and future work

In this paper, we present a hybrid business process to specify a data-oriented optimization business process, which includes declarative components focused on the data relationships between the activities, and imperative components focused on the control-flow perspective. In addition, we also show how an entirely imperative model can be created automatically using this hybrid model. It is carried out by means of the creation of a script task which solves a Constraint Optimization Problem, that can obtain the outcome data that optimize each instance of the process for customer requirements. Specifically, our focus is on problems formed by *white-box* activities, which means that the values of the output variables can be obtained in terms of the input values, and the pre and post-conditions, thereby rendering the execution of the activity unnecessary in the search for knowledge of the output of the activity. The use of the Constraint Programming paradigm enables the necessary input data values that optimize this outcome data to be ascertained. The approach presented is detailed and tested with a case study implemented in an open-source BPMS.

Further research could focus on the creation of the workflow of the subprocess when it is not included in the description of the model.

Acknowledgment

This work has been partially funded by the Ministry of Science and Technology of Spain (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

References

[1] R.S. Aguilar-Saven, Business process modelling: review and framework, Int. J. Prod. Econ. 90 (2) (2004) 129–149 ISSN: 0925-5273.
[2] A. Awad, M. Weidlich, M. Weske, Specification, verification and explanation of violation for data aware compliance rules, in: Proceedings of the 7th International Joint Conference on Service-Oriented Computing ICSOC-ServiceWave '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 500–515 ISBN: 978-3-642-10382-7.
[3] I. Barba, B. Weber, C.D. Valle, A.J. Ramírez, User recommendations for the optimized execution of business processes, Data Knowl. Eng. 86 (0) (2013) 61–84 ISSN: 0169-023X.
[4] D. Calvanese, G. De Giacomo, M. Montali, Foundations of data-aware process analysis: a database theory perspective, in: Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS, New York, NY, USA, 2013, pp. 1–12 ISBN: 978-1-4503-2066-5.
[5] M. Castellanos, N. Salazar, F. Casati, U. Dayal, M.-C. Shan, Predictive business operations management, in: S. Bhalla (Ed.), Databases in Networked Information Systems, Lecture Notes in Computer Science, vol. 3433, Springer, Berlin Heidelberg, 2005, pp. 1–14 ISBN: 978-3-540-25361-7.
[6] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the IJCAI, Morgan Kaufmann, 1991, pp. 331–340 ISBN: 1-55860-160-0.
[7] Team choco solver, 2014, Choco 3.1.1. http://www.emn.fr/z-info/choco-solver/ (accessed 23.10.15).
[8] Bonita community, 2012, Bonita Open Solution. http://www.bonitasoft.org/ (accessed 23.10.15).

[9] R. Conforti, M. de Leoni, M.L. Rosa, W.M. van der Aalst, Supporting risk-informed decisions during business process execution, in: Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE'13), Springer, Berlin Heidelberg/Valencia, Spain, 2013, pp. 116–132 ISBN: 978-3-642-38708-1.

[10] R. Dechter, Constraint Processing, Elsevier Morgan Kaufmann, 2003 ISBN: 978-1-55860-890-0.

[11] G. De Giacomo, M. Dumas, F.M. Maggi, M. Montali, Declarative process modeling in BPMN, in: Advanced Information Systems Engineering, in: Lecture Notes in Computer Science, vol. 9097, Springer, Berlin Heidelberg, 2015, pp. 80–100 ISBN: 978-3-319-19068-6.

[12] R. De Masellis, F.M. Maggi, M. Montali, Monitoring data-aware business constraints with finite state automata, in: Proceedings of the 2014 International Conference on Software and System Process. ICSSP 2014. ACM, New York, NY, USA, 2014, pp. 134–143 ISBN: 978-1-4503-2754-1.

[13] J. Fabra, V.D. Castro, P. Álvarez, E. Marcos, Automatic execution of business process models: exploiting the benefits of model-driven engineering approaches, J. Syst. Softw. 85 (3) (2012) 607–625 ISSN: 0164-1212.

[14] D. Fahland, D. Lubke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, S. Zugal, Declarative versus imperative process modeling languages: the issue of understandability, in: Enterprise, Business-Process and Information Systems Modeling, in: Lecture Notes in Business Information Processing, vol. 29, Springer Berlin Heidelberg, 2009, pp. 353–366 ISBN: 978-3-642-01861-9.

[15] D. Fahland, J. Mendling, H. Reijers, B. Weber, M. Weidlich, S. Zugal, Declarative versus imperative process modeling languages: The issue of maintainability, in: Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 43, Springer, Berlin Heidelberg, 2010, pp. 477–488 ISBN: 978-3-642-12185-2.

[16] E. Foundations, Epsilon (2014a). https://www.eclipse.org/epsilon/(accessed 24.04.14).

[17] E. Foundations, Epsilon Transformation Language (2014b). http://www.eclipse.org/epsilon/doc/etl/(accessed 24.10.14).

[18] S. Goedertier, J. Vanthienen, Designing compliant business processes with obligations and permissions, in: Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 4103, Springer, Berlin Heidelberg, 2006a, pp. 5–14 ISBN: 978-3-540-38444-1.

[19] S. Goedertier, J. Vanthienen, Business Rules for Compliant Business Process Models, in: Proceedings of the 9th International Conference on Business Information Systems, BIS Klagenfurt, Austria, 2006b, pp. 558–572 ISBN: 3-88579-179-X.

[20] S. Goedertier, J. Vanthienen, Em-bra2ce v0.1: A vocabulary and execution model for declarative business process modeling, in: Department of Decision Sciences and Information Management – KBI, 2007.

[21] S. Goedertier, C. Mues, J. Vanthienen, Specifying process-aware access control rules in SBVR, in: A. Paschke, Y. Biletskiy (Eds.), Advances in Rule Interchange and Applications, Lecture Notes in Computer Science, vol. 4824, Springer, Berlin Heidelberg, 2007, pp. 39–52 ISBN: 978-3-540-75974-4.

[22] S. Goedertier, J. Vanthienen, Declarative process modeling with business vocabulary and business rules, in: Proceedings of the 2007 OTM Confederated International Conference on the Move to Meaningful Internet Systems – Volume Part I, OTM'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 603–612 ISBN: 978-3-540-76887-6.

[23] M.T. Gómez-López, R.M. Gasca, Run-time auditing for business processes data using constraints, in: Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 66, Springer Berlin Heidelberg, 2010, pp. 146–157 ISBN: 978-3-642-20510-1.

[24] 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: Proceedings of the 25th International Conference on Information System Development. ISD 2011, Springer, 2011, pp. 15–25 ISBN: 978-1-4614-4950-8.

[25] S.-M. Huang, Y.-T. Chu, S.-H. Li, D.C. Yen, Enhancing conflict detecting mechanism for web services composition: A business process flow model transformation approach, Inf. Softw. Technol. 50 (11) (2008) 1069–1087 ISSN: 0950-5849.

[26] A.J. Ramírez, I. Barba, C.D. Valle, B. Weber, Generating multi-objective optimized business process enactment plans, in: Proceedings of the 25th International Conference on Advanced Information Systems Engineering. CAISE '13, 2013, pp. 99–115 ISBN: 978-3-642-38708-1.

[27] D. Knuplesch, L.T. Ly, S. Rinderle-Ma, H. Pfeifer, P. Dadam, On enabling data-aware compliance checking of business process models, in: Conceptual Modeling - ER, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010, pp. 332–346 ISBN: 978-3-642-16372-2.

[28] R.A. Kowalski, M.J. Sergot, A logic-based calculus of events, New Gener. Comput. 4 (1) (1986) 67–95 ISSN: 0288-3635.

[29] A. Krzysztof, Principles of Constraint Programming, Ed., Cambridge University Press, New York, NY, USA, 2003 ISBN: 978-0-521-12549-9.

[30] B. Li, J. Iijima, A Hybrid Approach for Business Process Verification, in: Research and Practical Issues of Enterprise Information Systems II, vol. 254, IFIP The International Federation for Information Processing, 2008, pp. 1–9 ISBN: 978-1-4757-0563-8.

[31] L.T. Ly, S. Rinderle-Ma, P. Dadam, Integration and verification of semantic constraints in adaptive process management systems, Data Knowl. Eng. 64 (1) (2008) 3–23 ISSN: 0169-023X.

[32] L.T. Ly, S. Rinderle-Ma, P. Dadam, Design and verification of instantiable compliance rule graphs in process-aware information systems, in: Proceedings of the 22th International Conference on Advanced Information Systems Engineering. CAISE '10, 2010, pp. 9–23 ISBN: 978-3-642-13093-9.

[33] L.T. Ly, S. Rinderle-Ma, D. Knuplesch, P. Dadam, Monitoring business process compliance using compliance rule graphs, in: Proceedings of the 19th International Conference on Cooperative Information Systems (CoopIS 2011), in: Volume 7044 of LNCS, Springer, 2011, pp. 82–99 ISBN: 978-3-642-25108-5.

[34] F.M. Maggi, M. Montali, M. Westergaard, W. van der Aalst, Monitoring business constraints with linear temporal logic: an approach based on colored automata, in: Proceedings of the 9th International Conference on Business Process Management, Springer-Verlag, 2011, pp. 132–147 ISBN: 978-3-642-23058-5.

[35] F.M. Maggi, M. Westergaard, M. Montali, W. van der Aalst, Runtime Verification of LTL-Based Declarative Process Models, in: Proceedings of RV, in: LNCS, Springer-Verlag, 2011c ISBN: 978-3-642-29859-2.

[36] F.M. Maggi, M. Dumas, L.G.-B. nuelos, M. Montali, Discovering data-aware declarative process models from event logs, in: F. Daniel, J. Wang, B. Weber (Eds.), Business Process Management, Lecture Notes in Computer Science, vol. 8094, Springer, Berlin Heidelberg, 2013a, pp. 81–96 ISBN: 978-3-642-40175-6.

[37] F.M. Maggi, C.D. Francescomarino, M. Dumas, C. Ghidini, Predictive Monitoring of Business Processes, CoRR abs/1312.4874 (2013b) ISBN: 978-1-4673-2318-5.

[38] F.M. Maggi, T. Slaats, H.A. Reijers, The Automated Discovery of Hybrid Processess, in: Business Process Management, in: Lecture Notes in Computer Science, vol. 8659, Springer Berlin Heidelberg, 2014, pp. 392–399 ISBN: 978-3-319-10171-2.

[39] M. Le, B. Gabrys, D. Nauck, A hybrid model for business process event and outcome prediction, Expert Syst. (2014) ISSN: 1468-0394.

[40] F.M. Maggi, M. Westergaard, W.M.P. van der Aalst, F. Staff, M. Pesic, H. Schonenberg, Declare tool (2014) (accessed 24.02.14).

[41] A. Meyer, L. Pufahl, D. Fahland, M. Weske, Modeling and enacting complex data dependencies in business processes, in: Business Process Management, in: Lecture Notes in Computer Science, vol. 8094, Springer Berlin Heidelberg, 2013, pp. 171–186 ISBN: 978-3-642-40175-6.

[42] A. Meyer, S. Smirnov, M. Weske, Data in business processes, EMISA Forum 31 (3) (2011) 5–31.

[43] M. Montali, Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach, in: Lecture Notes in Business Information Processing, vol. 56, Springer, 2010 ISBN: 978-3-642-14537-7.

[44] M. Montali, F. Chesani, P. Mello, F.M. Maggi, Towards data-aware constraints in declare, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, ACM, 2013, pp. 1391–1396 ISBN: 978-1-4503-1656-9.

[45] M. Montali, F.M. Maggi, F. Chesani, P. Mello, W.M.P. van der Aalst, Monitoring business constraints with the event calculus, ACM Trans. Intell. Syst. Technol. 5 (1) (2014) 17:1–17:30 ISSN: 2157-6904.

[46] M. Montali, Specification and Verification of Declarative Open Interaction Models: A Logic-Based Approach, University of Bologna, 2009 (Ph.d. thesis).

[47] B.V. Nuffelen, A.C. Kakas, A-system: declarative programming with abduction, in: Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, , LPNMR 2001, vol. 2173, Springer, Vienna, Austria, September 17-19, 2001,, pp. 393–396 ISBN: 3-540-42593-4.

[48] OMG, Business Process Model and Notation (BPMN) Version 2.0. Object Management Group Standard (2011).

[49] L. Parody, M.T. Gómez-López, R.M. Gasca, Extending BPMN 2.0 for modelling the combination of activities that involve data constraints, in: J. Mendling, M. Weidlich (Eds.), BPMN, Lecture Notes in Business Information Processing, vol. 125, Springer, 2012a, pp. 68–82 ISBN: 978-3-642-33154-1.

[50] L. Parody, M.T. Gómez-López, R.M. Gasca, A.J. Varela-Vaca, Improvement of optimization agreements in business processes involving web services, Commun. IBIMA, 2012 (2012b), doi:10.5171/2012.959796.

[51] M. Pesic, W.M.P. van der Aalst, A declarative approach for flexible business processes management, in: J. Eder, S. Dustdar (Eds.), Business Process Management Workshops, Lecture Notes in Computer Science, vol. 4103, Springer, 2006, pp. 169–180 ISBN: 978-3-540-38444-1.

[52] M. Pesic, Constraint-Based Workflow Management Systems: Shifting Control to Users, Technische Universiteit Eindhoven, 2008 (Ph.D. thesis).

[53] M. Pesic, H. Schonenberg, W.M.P. van der Aalst, Declarative workflow, in: A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, N. Russell (Eds.), Modern Business Process Automation, Springer, 2010, pp. 175–201 ISBN: 978-3-642-03121-2.

[54] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, H.A. Reijers, Imperative versus declarative process modeling languages: an empirical investigation, in: Business Process Management Workshops (1), in: Lecture Notes in Business Information Processing, vol. 99, Springer, 2011, pp. 383–394 ISBN: 978-3-642-28107-5.

[55] M. Reichert, B. Weber, Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies, Springer, 2012 ISBN: 978-3-642-30408-8.

[56] H.A. Reijers, T. Slaats, C. Stahl, Declarative modelingan academic dream or the future for BPM?, in: F. Daniel, J. Wang, B. Weber (Eds.) Business Pprocess Management 2013, Lecture Notes in Business Information Processing, vol. 8094, Springer, 2013, p. 307322 ISBN: 978-3-642-40175-6.

[57] W.D. Roover, F. Caron, J. Vanthienen, A prototype tool for the event-driven enforcement of SBVR business rules, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops (1), Lecture Notes in Business Information Processing, vol. 99, Springer, 2011, pp. 446–457 ISBN: 978-3-642-28107-5.

[58] G. Rummler, A. Brache, Improving Performance: How to Manage the White Space on the Organization Chart, Jossey-Bass, San Francisco, CA, 1995 ISBN: 978-1-118-14370-4.

[59] I. Rychkova, G. Regev, A. Wegmann, High-level design and analysis of business processes: the advantages of declarative specifications, in: O. Pastor, A. Flory, J.-L. Cavarero (Eds.), Proceedings of the RCIS, IEEE, 2008a, pp. 99–110 ISBN: 978-1-4244-1677-6.
[60] I. Rychkova, G. Regev, A. Wegmann, Using declarative specifications in business process design, Int. J. Comput. Sci. Appl. 5 (3b) (2008b) 45–68.
[61] I. Rychkova, S. Nurcan, Towards adaptability and control for knowledge-intensive business processes: Declarative configurable process specifications, in: Proceedings of the HICSS, 2011a, pp. 1–10 ISBN: 978-1-4244-9618-1.
[62] I. Rychkova, S. Nurcan, The old therapy for the new problem: declarative configurable process specifications for the adaptive case management support, in: Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 66, Springer Berlin Heidelberg, 2011b, pp. 420–432 ISBN: 978-3-642-20510-1.
[63] S.W. Sadiq, M.E. Orlowska, W. Sadiq, Specification and validation of process constraints for flexible workflows, Inf. Syst, 30 (5) (2005) 349–378 ISSN: 0306-4379.
[64] S. Sadiq, M. Orlowska, J. Lin, W. Sadiq, Quality of service in flexible workflows through process constraints, in: C.-S. Chen, J. Filipe, I. Seruca, J. Cordeiro (Eds.), Enterprise Information Systems VII, Springer Netherlands, 2006, pp. 187–195 ISBN: 978-1-4020-5323-8.
[65] OMG, Semantics of business vocabulary and business rules (SBVR) (2008) (accessed 24.02.14).
[66] H. Schonenberg, B. Weber, B.F. van Dongen, W.M.P. van der Aalst, Supporting flexible processes through recommendations based on history, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), BPM, Lecture Notes in Computer Science, vol. 5240, Springer, 2008, pp. 51–66 ISBN: 978-3-540-85758-7.
[67] V. Torres, V. Pelechano, Building business process driven web applications, in: S. Dustdar, J.L. Fiadeiro, A.P. Sheth (Eds.), Business Process Management, Lecture Notes in Computer Science, vol. 4102, Springer, 2006, pp. 322–337 ISBN: 3-540-38901-6.
[68] W.M.P.V.d. Aalst, M. Stoffele, J. Wamelink, Case handling in construction, Autom. Constr. 12 (3) (2003) 303–320 ISSN: 0926-5805.
[69] W.M.P. van der Aalst, M. Weske, D. Grübauer, Case handling: A new paradigm for business process support, Data Knowl. Eng. 53 (2) (2005) 129–162 ISSN: 0169-023X.
[70] W.M.P. van der Aalst, C. Stahl, A Petri Net-Oriented Approach, The MIT Press, 2011 ISBN: 978-0-262-01538-7.
[71] R.J. Wallace, Directed arc consistency preprocessing, Constraint Processing, Selected Papers, Springer-Verlag, London, UK, 1995, pp. 121–137 ISBN: 3-540-59479-5.
[72] B. Weber, S.W. Sadiq, M. Reichert, Beyond rigidity - dynamic process lifecycle support, Comput. Sci. R&D 23 (2) (2009) 47–65 ISSN: 1865-2034.
[73] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer, 2007 ISBN: 978-3-540-73521-2.
[74] M. Weske, Formal foundation and conceptual design of dynamic adaptations in a workflow management system, in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001, 2001, p. 10. ISBN: 0-7695-0981-9.
[75] M. Westergaard, Better algorithms for analyzing and enacting declarative workflow languages using ltl, in: S. Rinderle-Ma, F. Toumani, K. Wolf (Eds.), Business Process Management, Lecture Notes in Computer Science, vol. 6896, Springer Berlin Heidelberg, 2011, pp. 83–98 ISBN: 978-3-642-23059-2.
[76] X. Zhou, S. Sadiq, S.H. Tao, X. Li, M.A. Sharaf, Z. Huang, K. Zheng, J. Hunter, P. Green, M. Indulska, Data Centric Research at The University of Queensland, Sigmod Rec. 42 (3) (2013) 63–68 ISSN: 0163-5808.
[77] S. Zugal, P. Soffer, J. Pinggera, B. Weber, Expressiveness and understandability considerations of hierarchy in declarative business process models, in: I. Bider, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, S. Wrycza (Eds.), Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing, vol. 113, Springer Berlin Heidelberg, 2012, pp. 167–181 ISBN: 978-3-642-31072-0.