Data-Oriented Declarative Language for Optimizing Business Processes

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

Abstract There is a signifi cant number of declarative languages to describe business processes. They tend to be used when business processes need to be flexible and adaptable, being not possible to use an imperative description. Declarative languages in business process have been traditionally used to describe the order of activities, specifi cally the order allowed or prohibited. Unfortunately, none of them is worried about a declarative description of exchanged data between the activities and how they can infl uence the model. In this paper, we analyse the data description capacity of a variety of declarative languages in business processes. Using this analysis, we have detected the necessity to include data exchanged aspects in the declarative descriptions. In order to solve the gap, we propose a Data-Oriented Optimization Declarative LanguagE, called DOODLE, which includes the process requirements referred to data description, and the possibility to include an optimization function about the process output data.

Keywords Business Processes • Declarative Language • Data-Oriented Optimization

5.1 Introduction

A business process, henceforth referred to as BP, consists of a set of activities that are performed in coordination within an organizational and technical environment. These activities jointly perform a business goal [1]. In management theory during the last years, a process-oriented perspective has been considered the shell in organizational (re)structuring. Nowadays, organizations still experience difficulties in applying this process-oriented perspective to the design and maintenance of their information systems. Currently, the deployment of more complex systems has put forth new requirements for flexible and dynamic specification. Several languages propose an imperative representation of business processes, whose specification allows business experts to describe relationships between activities, and to transform the process into an executable model. Therefore, an imperative description defines exactly how activities have to be performed, and how to handle the data-flow. However, one of the disadvantages of imperative languages comes from the use of unsuitable information for computer systems, since they do not provide flexible and adaptable business processes.

A declarative representation takes into account the business concerns that govern the BP. A BP may be exposed to different environments and subjected to many conditions in which the order of activities, or the data exchanged, cannot always be described at design time or in an easy way. This is the reason why several authors have proposed languages to define BP as declarative models, since sometimes the process cannot be completely defined at design time. One of the reasons why it is not possible to create an imperative model at design time, is related to the data that flow through the process. Depending on the data instantiated, the creation of one model or another could be better, since, for example, the model will influence the selection of the best data input of the activities to satisfy the process requirements. The role of data in declarative languages has not been very relevant, mostly limited to describe the execution or not of an activity, depending on the value of a variable of the data-flow. In this paper, we present a new point of view of declarative language focused on data, where we highlight the significance of the information that flows through the process and between the activities to reach an optimal model according to the user requirements.

Sometimes, the user of an application supported by a business process has to decide about the values to introduce at runtime, for example, the dates in the organization of a trip (booking flights, hotel room and renting a car). Frequently, if the user can choose between different dates in order to minimize the price, (s)he needs to search by hand the combinations of dates with the activities this mean searching for flights, hotel rooms and car rental. For this example, the model is known, being possible to execute the activities in a parallel manner. However, the process goal is to book a trip, and the objective function is to minimize the price of the trip. Then, in order to achieve the user requirements (to minimize the sum of the prices for the services, for a given set of dates), the best combination of data input for the activities needs to be found. To the best of our knowledge, none of the declarative languages permit the inclusion of data output optimization in this sense. For this reason, we propose a data-oriented decision language, called DOODLE. Although there is already a declarative language called DOODLE, presented by Cruz in [2], it is a visual and declarative language for object-oriented databases and our proposal is focused on a definition of a business process in a declarative way. Since DOODLE is oriented to data perspective, when in this paper we use the term ‘optimize the process’, it means that there is a function to be optimized and where the data-flow involved in the process are related with the aim to optimize the data output.

To meet this challenge, we have analysed the existing declarative languages and how they have addressed data management, and the features that have not yet been analysed.

The rest of the paper is organized as follows: Sect. 5.2 motivates and explains, through an illustrative example, the necessity in some cases to find the data input values to optimize the process execution. Section 5.2 introduces a motivating example where a declarative description oriented to data is necessary. Section 5.3 details the proposed language based on the description of a declarative subprocess that can be combined with an imperative description, such as BPMN. Section 5.4 studies some of the most relevant proposals of declarative languages and their contributions to data management. The motivating example described by means of this language has been included and a comparison with the studied languages is presented. And finally, conclusions are drawn and future work is proposed in Sect. 5.5.

5.2 On-Line Book Store: A Motivating Example

In this section, we introduce an example to motivate the necessity to include in declarative languages some aspects related to data that have influence in the model. The example is based on a sale and delivery process that has been used in many papers before [3–5] and [6]. The example is the on-line Book Store (BS), that represents a company that collaborates with two services in order to sell and deliver books (see Fig. 5.1). Both services inform the customer about the final price of buying and delivering a number of units of a book. However, it could be cheaper to buy this quantity of books in different packages by obtaining a discount. For example, if there is a discount depending on the number of units for a maximum, or the price of shipping depends on both the weight and volume of the boxes to send. In this case, it is cheaper to send two small boxes (e.g. 8€ each one) than a bigger one (e.g. 25€). Another example is if the customer wants to buy 5 units of a book, the cheapest option could be to buy them in two packages (3 + 2), since there is an offer “buy 3 pay 2”, and although two deliveries are paid, the delivery cost is increased considerably due to the weight and volume. The sale terminates when the books are delivered to the customer and the payment is made.

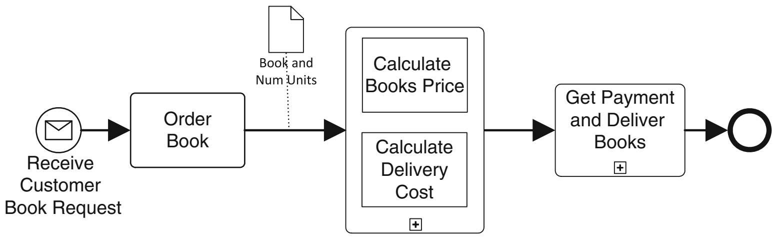


Fig. 5.1 On-line book store example

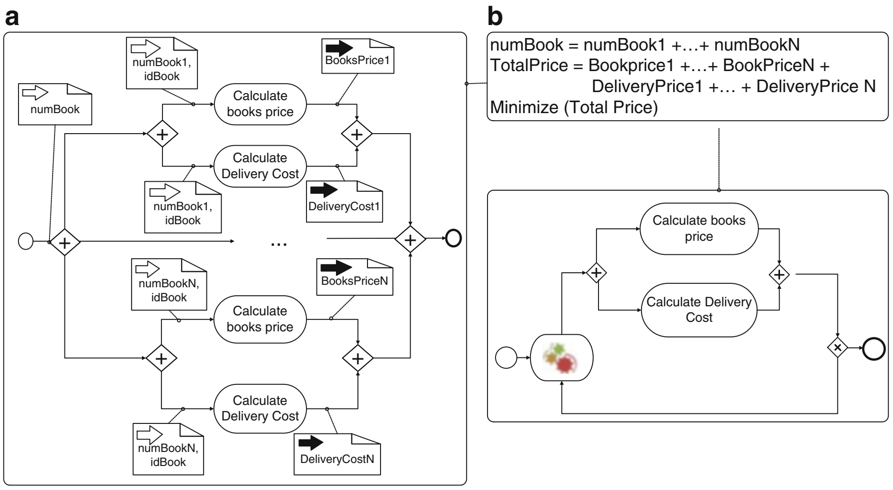


Fig. 5.2 Imperative models for the on-line book store example. (**a**) Deployed imperative model. (**b**) Imperative model with implementation of tasks

In order to model the business process which support the BS example including the requirement to minimize the final price, neither imperative nor declarative languages can be used. For imperative languages, Fig. 5.2 depicts two possibilities of implementation to satisfy the constraints, which are also included in the figure. On the one hand, Fig. 5.2a presents a possible imperative model in BPMN to describe the BS example. The model has several activities to execute the purchase and the delivery for each package, but the problem is that the *numBooki* variables are unknown, even at runtime, since they are determined to minimize the total cost in each instance. Moreover, the number of activities to buy and deliver the books are unknown at design time. The difficulty in this case is that the specific values of the variables and the number of activities executed to minimize the objective function are unknown at design time. On the other hand, another solution could be that shown in Fig. 5.2b: various executions of the activities can be made by means of a loop, and the values of the *numBooki* variables will be determined programmatically in a new activity included in the model. The problem with this solution is the significant difficulty in implementing this activity, being necessary to delegate their programming to a computer science expert, not to a business process expert.

In relation to the existing declarative languages, as is analysed in Sect. 5.4, to the best of our knowledge, the current declarative languages are centred on the declarative description of the order of activities. However, none of them includes data input and output of the activities to optimize the object obtained from the business process.

5.3 DOODLE: Data-Oriented Optimization Declarative LanguagE

In order to include the optimization function and the constraints into the business process model, we have defined a declarative language called DOODLE (Data-Oriented Optimization Declarative LanguagE). This language combines the imperative description of BPMN with a declarative description of the parts of the process that need more flexibility (declarative subprocesses). The declarative description includes the data subprocess, data activities, objective function and the numerical constraints that let the designer describe the possible data values in a declarative way. These numerical constrains are defined by the following grammar where `Variable` and `Value` can be defined as Integer, Natural or Float domain:

```
Constraint := Atomic_Constraint BOOL_OP Constraint
| Atomic_Constraint
| '¬'Constraint
BOOL_OP:= '∨' |'∧'
Atomic_Constraint:= function PREDICATE function
function:= Variable FUNCTION_SYMBOL function
| Variable
    | Σ Variable
    | Value
  PREDICATE:= '=' |'<' |'≤' |'>' |'≥'
  FUNCTION_SYMBOL:= '+' |'−' |'∗'
```

These constraints make it easier and more precise when handling numeric data (that can be represented as variables) that represent relations between variables.

In order to introduce the language, we have divided the description into two parts: (i) the internal description of the components associated with the activities of the declarative subprocess (Table 5.1), and (ii) the external description of the declarative subprocess, that implies the relation of the activities with subprocess data input and output also with constraints (Table 5.2). The language proposes to describe the order of the activities using imperative or declarative languages, depending on the necessity of the process. For the BS example, a parallel execution is possible, then it can be described imperatively.

In order to understand the example better, the language has been used to model the BS problem, as shown in Fig. 5.3. In order to transform this declarative model into an imperative model that supports any value of input variables of the process (idBook and number Of Books for example), we propose the use of Constraint Programming paradigm (explained in Sect. 5.4.1) [7] and domain local analysis of the variables [8]. The created model will depend on the knowledge of the relationship between input and output data of the activities, specifically, if this relationship

Table 5.1 Internal components associated to the activities of the declarative subprocess

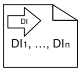
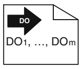
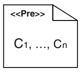
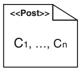
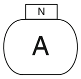
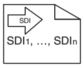
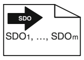
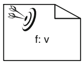
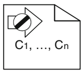
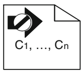
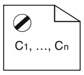
Symbol	Name	Description	Parameters
	Data input (DI) of the activity	Set of data input of each activity	List of variables
	Data output (DO) of the activity	Set of data output of each activity	List of variables
	Precondition	Set of constraints that represents the values of the DI that satisfy the execution of the activity	Numerical constraints
	Postcondition	Set of constraints that represents the values of the DO that are satisfied after the execution of the activity	Numerical constraints
	Repetition of an activity A	Representation of the number of times that an activity can be executed. The value can be numeric (e.g. 5), or symbolic (e.g. N)	Integer or string

Table 5.2 External components of the declarative subprocess

Symbol	Name	Description	Parameters
	Subprocess data input (SDI)	Data input of the subprocess that describe the user requirement in each process execution	List of variables
	Subprocess data input (SDI)	Data input of the subprocess that describe the user requirement in each process execution	List of variables
	Objective function	An optimization function in terms of a data	Minimize or maximize and the objective variable
	ID input constraints (IC)	Set of constraints that relates the SDI with the DI of each activity of the subprocess	Numerical constraints
	ID output constraints (OC)	Set of constraints that relates the SDO with the DO of each activity of the subprocess	Numerical constraints
	Internal constraints	Set of constraints that relates the DI and DO of the activities among them	Numerical constraints

can be known with or without executing the activities. For the BS example, the *totalPrice* can only be known if the activities are executed, although if the pre and post-conditions of the activities could relate input and outputs, the execution of the activities would be not necessary.

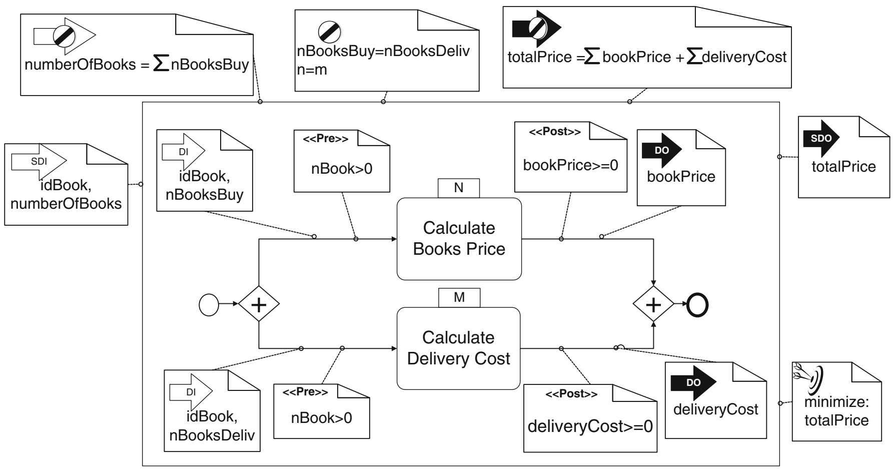


Fig. 5.3 Example of buy books described using DOODLE

5.4 Related Work

There are many languages that enable the description of business processes in a declarative way. Generally, the common idea of declarative business process modelling is to model a process as a trajectory in a state space, and declarative constraints are used to define the valid movements in that state space [9]. Therefore, the differences between declarative process languages can, in part, be understood as a different perception of the meaning of 'state'. Analysing the different proposals, we have found some characteristics that we consider interesting to be compared. The characteristics are studied in Sect. 5.4.1 and compared in Sect. 5.4.2.

5.4.1 Declarative Language Characteristics

Analysing the different proposals related to declarative languages, the main characteristics that we think should be analysed and compared are:

- **Formalism for reasoning:** The proposals use different formalism for reasoning. Sometimes, although we show the most relevant in each case, they can combine more than one and be improved with made-to-measure algorithms. Only the most relevant have been included in the paper due to space limitations.
  - **Linear Temporal Logic (LTL).** As demonstrated by Chomicki [10], Bacchus and Kabanza [11] and Pesic and van der Aalst [12], LTL expressions can be used to represent desirable or undesirable patterns within a history of events. LTL is a modal temporal logic that allows the expression of temporal constraints

on infinite paths within a state space. LTL formula can be evaluated by obtaining the Büchi automaton that is equivalent to the formula and checks whether a path corresponds to the automaton. Unfortunately most LTL checking algorithms assume infinite paths and construct non-deterministic automata [12]. Another disadvantage is that LTL does not allow the expression of the effect that results from a particular transition in a state space. For these reasons, it is not evident to express a goal state in LTL nor to construct automata for planning an execution scenario to obtain a goal state [11] as is needed in an optimization function.

- **The Event Calculus.** In first-order logic there is a formalism that elegantly captures the time-varying nature of facts, the events that have taken place at given time points and the effect of these events reflecting on the state of the system. This formalism is called the Event Calculus. The Event Calculus, introduced by Kowalski and Sergot [13], is a logic programming formalism to represent and reason about the effect of events on the state of a system. The Event Calculus is appealing for several reasons, as it builds on a first-order predicate logic framework, for which efficient reasoning algorithms exist. In addition the Event Calculus not only has the ability to deductively reason about the effects of the occurrence of events (leading to the coming into existence of fluency or the ceasing to hold), most importantly, it also has the ability to reason abductively. Abductive reasoning over the event calculus has been shown to be equivalent to planning. In particular, abductive reasoning produces a sequence of transitions (denoted by events) that must happen for a particular instance to hold in the future [14–16].
- **Constraint Programming (CP).** Constraint Programming [17] is a paradigm that permits the description of the model by means of the variables and the constraints that relate the variables. The model is called a Constraint Satisfaction Problem (CSP), that represents a reasoning framework consisting of variables, domains and constraints. Formally, it is defined as a tuple (X, D, C), where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \ldots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \ldots, C_m\}$ is a set of constraints. Each constraint C_i is defined as a relation R on a subset of variables $V = \{x_i, x_j, \ldots, x_l\}$, called the *constraint scope*. The relation R may be represented as a subset of the Cartesian Product $d(x_i)$ x $d(x_j)$ x … x $d(xl)$. A constraint $Ci = (V_i,R_i)$ simultaneously specifies the possible values of the variables in V in order to satisfy R. Let $V_k = \{x_{k1}, x_{k2}, \ldots, x_{kl\}}$ be a subset of X, and an l-tuple $(x_{k1}, x_{k2,} \ldots, x_{kl)}$ from $d(x_{k1}), d(x_{k2}), \ldots, d(x_{kl})$ can therefore be called an *instantiation* of the variables in V_k. An instantiation is a solution if and only if it satisfies the constraints C.

* **Imperative and Declarative:** This is the capacity of a language to describe imperative and declarative aspects in the same model, since sometimes a part of the process is completely unknown, and other parts are totally known.

- **Use of the language:** The existing proposals that we have analysed are focused on different objectives: *Validation* of the model for a trace of events, *Construction* of automatons to generate a possible sequence of activities, or *Assistance* to the user to decide which is the best activity to execute at runtime.
- **Data perspective:** The possibility to include the values of the data-flow variables in the rules that describe the declarative model.
- **Pre and Post-condition:** The inclusion of a description of the system before and after it is instantiated by means of pre and post-conditions. This is a relevant aspect since it allows the modeller to describe the data before and after the process execution, without the inclusion of details about the internal description of the process.
- **Optimization Function:** The possibility to include an optimization function in the declarative description that is taken into account in the model.

5.4.2 Analysis of Declarative Languages

Some of the most important declarative languages have been included and compared in this section.

- **Pocket of flexibility.** This solution is based on constraint specification of the business process workflow. The constraint specification framework [18] represents the workflow as a directed graph where there are two types of nodes: activity nodes and coordinator nodes. In the framework, it is possible to combine activities whose relation is known with activities whose relation is unknown (called pocket of flexibility) that include a set of constraints for concretizing the pocket with a valid composition of work-flow fragments. It includes different types of constraints (Serial, Order, Fork, Inclusion, Exclusion, Number of executions for activity or in parallel). The constraints relate the number of times that each activity can be executed and the order and/or mandatory execution of each activity. The proposal defines a set of algorithms to find possible discrepancies between the constrains that describe the process and an instance. The implementation is based on a made-to-measure algorithm that uses the graph to represent the constraints. The implementation has been included in the prototype called *Chameleon*. The data aspect has not been included in this proposal.
- **DeCo.** Irina Rychkova et al. in [19], [4] and [20] presented a declarative BP specification language that enables designers to describe the actions that a business process needs to contain, but not where their specific sequence can be postponed to the instance time. They improve the alignment of the BP with the business strategy of an organization by giving a synthesis of a set of business processes (abstracting the control flow), while maintaining a rigorous relationship with the detailed process. These specifications complement the traditional

(imperative) business process model by specifying the process independently from a particular environment, (e.g. a process can be executed). This technique includes checking the conformance of the imperative and the declarative specifications, using the case handling paradigm [21]. For every action of the working object they define a precondition and a postcondition. A precondition specifies a set of states where the action can be executed, and postcondition specifies the possible set of states after the action was executed. The pre and postcondition 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, as different imperative description for the same declarative descriptions are possible. Thereby this proposal focuses on the problem from the working object point of view, and data values is one of the analysis.

- **Compliance Rule Graphs.** The Compliance Rule Graphs (CRGs) [5, 22, 23] focus their challenge 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 be included as well. The proposal verifies the correctness of the process analysing the compliance rules and the events monitored. The description of the order of activities can be enriched including conditions to the rules that will be satisfied or not depending on the data value for each instance. The analysis is done using pattern matching mechanisms, and is included in a prototype called SeaFlow.
- **Em-Bra^{2}Ce.** The Enterprise Modeling using Business Rules, Agents, Activities, Concepts and Events (Em-Bra^{2}Ce) Framework [24, 25] presents a declarative language based on SBVR (Semantics Of Business Vocabulary And Business Rules) to describe the vocabulary of the process, and an execution model to represent the control flow perspective based on Colored Petri Nets. The use of SVBR allows the description of data aspects in the business process that can be included in the ECA (Event Condition Action) rules, used as a pattern to write the rules.
- **Penelope.** The language Penelope (Process ENtailment from the ELicitation of Obligations and PErmissions) [6] expresses temporal rules about the obligations and permissions in a business interaction using Deontic logic. This language is supported by an algorithm to generate compliant sequence-flow-based process models that can be used in business process design. This language uses the Event Calculus to model the effects of performing activities with respect to the coming into existence of temporal deontic assignments. The only type of data that is included in the definition is related to the execution time of the activities, but the data managed during each instance is not an object of the proposal.
- **ConDec.** The ConDec [12] language was designed for modelling and enacting dynamic business processes. The language defines the involved activities in the process and the order relations between them. This order relation is expressed using LTL to represent desirable or undesirable patterns within a history of

events. However, LTL formulas are difficult to read due to the complexity of expressions. Therefore, the authors have defined a graphical syntax for some typical constraints that can be encountered in workflows. ConDec initially defined three groups of templates to make the definition of activity relations easier: (1) existence, (2) relation and (3) negation templates. An automaton can be built in accordance with the ConDec model, where the automaton can be used to validate a sequence of events. Declare tool [26] is a prototype of a workflow management, that supports the ConDec language. This tool has been used for frameworks such as *Mobucon* [27, 28] for runtime validation. This framework allows for the continuous verification of compliance with respect to a predefined constraint model. ConDec has been enlarged to include the resource perspective (ConDec-R) and the data-aware constraints in Declare, both analysed in the following items.

- **ConDec-R.** This is an extension of the ConDec language to include a description of the resources necessary during process execution. The implementation extension, called ConDec-R [29], assists the user by means of recommendations to achieve an optimized plan for one or multiple objectives [30]. In order to obtain the plan, a CP 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 to activity input and output data.
- **Data-aware Constraints in Declare.** This is an extension of the Declare framework [31] that permits the representation of the input, internal and output data of the activities in a declarative representation of the order of the activity. Event calculus has been used to formalize the language and to validate if the traces of events are in accordance with the declarative model. Although the data aspect is included, only input and output data relations between activities can be described.

Although all these declarative languages include some information about data, none of them include the data input and output of the activities with the aim to optimize the object obtained from the business process.

5.4.3 *Declarative Languages Comparative*

Compared with declarative languages, our proposal DOODLE includes data-oriented aspects in a declarative manner. It is done by means of a set of constraints that describe the data exchanged among the activities, when their relations cannot be defined explicitly at design time. The model and the reasoning framework use Constraint Programming, in order to infer the possible values of the data and achieve the optimization function at run-time (Table 5.3).

Table 5.3 Declarative languages comparative

	Formalism	Imper. and Decl.	Use of model	Data perspective	Pre and post	Opt. function
Sadiq	Graph theory	✓	Validation			
DeCo	First Order Logic	✓	Validation	✓	✓	
CRGs	Pattern matching		Validation	✓		
Em-Bra^{2}Ce	Color Petri Net	✓	Validation	✓		
Penelope	Event calculus	✓	Construction			
ConDec	LTL		Validation			
ConDec-R	Const. programming		Assistance	✓		✓
Data-aware	Event calculus		Validation	✓		
Doodle	Const. programming	✓	Constr. and Assist.	✓	✓	✓

5.5 Conclusions

In this paper, we have analysed some of the most relevant declarative languages in business processes. From this analysis we have detected that none of them permit the data declarative description in the business processes, and how it can influence in the obtained model. In this paper, a declarative language called DOODLE is described, which permits the description of the data exchanged among the activities of the process in a declarative way by means of constraints, and obtains an optimization of the business process objects.

For future work, we plan to enlarge the language to support more complex semantics and data relations. We also consider interesting the development of a framework for the transformation from the declarative model to an imperative model implemented in a Business Process Management Systems, and supporting different types of technologies in a transparent way for the business modeller.

References

1. Weske M (2007) Business process management: concepts, languages, architectures. Springer, Secaucus, NJ
2. Cruz IF (1992) Doodle: a visual language for object-oriented databases. SIGMOD Rec 21(2):71–80
3. Sadiq S, Governatori G, Namiri K (2007) Modeling control objectives for business process compliance. In: Proceedings of the 5th international conference on business process management, BPM'07. Springer, Berlin, pp 149–164
4. Rychkova I, Regev G, Wegmann A (2008) High-level design and analysis of business processes the advantages of declarative specifications. In: IEEE international conference on research challenges in information science, RCIS'08, Marrakech, Morocco. pp 99–110. ISBN: 978-1-4244-1677-6

5. Knuplesch D, Ly LT, Rinderle-Ma S, Pfeifer H, Dadam P (2010) On enabling data-aware compliance checking of business process models. In: 29th International conference on conceptual modeling, ER 2010, Vancouver, BC, Canada, vol 6412. Springer, Berlin, pp 332–346. ISBN: 978-3-642-16372-2
6. Goedertier S, Vanthienen J (2006) Designing compliant business processes with obligations and permissions. In: Business process management workshops, vol 4103. Springer, Berlin, pp 5–14. ISBN: 978-3-540-38444-1
7. Parody L, Gomez-Lopez MT, Gasca M (2013) Decision-making sub-process to obtain the optimal combination of input data in business processes. In: Proceedings of the IX Jormadas de Ciencia e Ingeniería de Servicios, JCIS '13, pp 17–31. ISBN: 978-84-695-8351-7
8. Parody L, Gomez-Lopez MT, Martinez Gasca R, Varela-Vaca AJ (2012) Improvement of optimization agreements in business processes involving web services. Communications of the IBIMA 2012, vol 2012, Article ID 959796, 15 pp. doi:10.5171/2012.959796
9. Bider I, Khomyakov M, Pushchinsky E (2000) Logic of change: semantics of object systems with active relations. Autom Softw Eng 7:9–37
10. Chomicki J (1995) Depth-bounded bottom-up evaluation of logic program. J Log Program 25(1):1–31
11. Bacchus F, Kabanza F (2000) Using temporal logics to express search control knowledge for planning. Artif Intell 116(1–2):123–191
12. Pesic M, van der Aalst WMP (2006) A declarative approach for flexible business processes management. In: Proceedings of the 2006 international conference on business process management workshops, BPM'06. Springer, Berlin, pp 169–180
13. Kowalski RA, Sergot MJ (1986) A logic-based calculus of events. New Generat Comput 4(1):67–95
14. Eshghi K (1988) Abductive planning with event calculus. In: Proceedings of the fifth international conference and symposium on logic programming, Seattle, Washington. MIT Press. pp 562–579. ISBN: 0-262-61056-6
15. Shanahan M (1997) Event calculus planning revisited. In: Steel S, Alami R (eds) ECP. Lecture notes in computer science, vol 1348. Springer, Heidelberg, pp 390–402
16. Nuffelen BV, Kakas AC (2001) A-system: declarative programming with abduction. In: Eiter T, Faber W, Truszczynski M (eds) LPNMR. Lecture notes in computer science, vol 2173. Springer, Heidelberg, pp 393–396
17. Rossi F, Beek PV, Walsh T (2006) Handbook of constraint programming (foundations of artificial intelligence). Elsevier, New York, NY
18. Sadiq SW, Orlowska ME, Sadiq W (2005) Specification and validation of process constraints for flexible workflows. Inform Syst 30(5):349–378
19. Rychkova I, Regev G, Wegmann A (2008) Using declarative specifications in business process design. IJCSA 5(3b):45–68
20. Rychkova I, Nurcan S (2011) Towards adaptability and control for knowledge-intensive business processes: declarative configurable process specifications. In: 44th Hawaii international conference on system sciences (HICSS). pp 1–10. ISSN: 1530-1605
21. van der Aalst WM, Weske M, Grnbauer D (2005) Case handling: a new paradigm for business process support. Data Knowl Eng 53:129–162
22. Ly LT, Rinderle S, Dadam P (2008) Integration and verification of semantic constraints in adaptive process management systems. Data Knowl Eng 64(1):3–23
23. Ly LT, Rinderle-Ma S, Knuplesch D, Dadam P (2011) Monitoring business process compliance using compliance rule graphs. In: 19th International conference on cooperative information systems (CoopIS 2011). LNCS, vol 7044. Springer, Heidelberg, pp 82–99
24. Goedertier S, Haesen R, Vanthienen J (2007) Em-bra^{2}ce v0. 1: a vocabulary and execution model for declarative business process modeling. FETEW Research Report KBI 0728, K.U. Leuven
25. Roover WD, Caron F, Vanthienen J (2011) A prototype tool for the event-driven enforcement of sbvr business rules. In: Daniel F, Barkaoui K, Dustdar S (eds) Business process management workshops (1). Lecture notes in business information processing, vol 99. Springer, Heidelberg, pp 446–457

26. Maggi FM, Westergaard M, van der Aalst W, Staff F, Pesic M, Schonenberg H. Declare tool. http://www.win.tue.nl/declare/
27. Maggi FM, Montali M, Westergaard M, van der Aalst W (2011) Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: Proceedings of BPM. LNCS. Springer, Heidelberg
28. Maggi F, Westergaard M, Montali M, van der Aalst W (2011) Runtime verification of LTL based declarative process models. In: Proceedings of RV. LNCS. Springer, Heidelberg
29. Barba I, Weber B, Valle CD, Ramrez AJ (2013) User recommendations for the optimized execution of business processes. Data Knowl Eng 86:61–84
30. Ramrez AJ, Barba I, Valle CD, Weber B (2013) Generating multi-objective optimized business process enactment plans. In: 25th International conference on advanced information systems engineering. CAISE'13. Lecture notes in computer science, vol 7908. pp 99–115. ISBN: 978-3-642-38708-1
31. Montali M, Chesani F, Mello P, Maggi FM (2013) Towards data-aware constraints in declare. In: Proceedings of the 28th annual ACM symposium on applied computing, SAC '13. ACM, Coimbra, Portugal, pp 1391–1396