Information and Software Technology 70 (2016) 140–154

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

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

A R T I C L E   I N F O

Article history:
Received 8 March 2014
Revised 23 October 2015
Accepted 24 October 2015
Available online 30 October 2015

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 experts 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 example, 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

* Corresponding author. Tel.: +34 954556234; fax: +34 954557139.
E-mail addresses: lparody@us.es (L. Parody), maytegomez@us.es (M.T. Gómez-López), gasca@us.es (R.M. Gasca).

http://dx.doi.org/10.1016/j.infsof.2015.10.007
0950-5849/© 2015 Elsevier B.V. All rights reserved.

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