Expert Systems with Applications 41 (2014) 5340–5352

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Conformance checking and diagnosis for declarative business process models in data-aware scenarios

Diana Borrego *, Irene Barba

University of Seville, Department of Computer Languages and Systems, Av Reina Mercedes S/N, 41012 Seville, Spain

ARTICLE INFO

Keywords:
Business process management
Process mining
Conformance checking
Diagnosis
Declarative business process models
Constraint programming

ABSTRACT

A business process (BP) consists of a set of activities which are performed in coordination in an organizational and technical environment and which jointly realize a business goal. In such context, BP management (BPM) can be seen as supporting BPs using methods, techniques, and software in order to design, enact, control, and analyze operational processes involving humans, organizations, applications, and other sources of information. Since the accurate management of BPs is receiving increasing attention, conformance checking, i.e., verifying whether the observed behavior matches a modelled behavior, is becoming more and more critical. Moreover, declarative languages are more frequently used to provide an increased flexibility. However, whereas there exist solid conformance checking techniques for imperative models, little work has been conducted for declarative models. Furthermore, only control-flow perspective is usually considered although other perspectives (e.g., data) are crucial. In addition, most approaches exclusively check the conformance without providing any related diagnostics. To enhance the accurate management of flexible BPs, this work presents a constraint-based approach for conformance checking over declarative BP models (including both control-flow and data perspectives). In addition, two constraint-based proposals for providing related diagnosis are detailed. To demonstrate both the effectiveness and the efficiency of the proposed approaches, the analysis of different performance measures related to a wide diversified set of test models of varying complexity has been performed.

1. Introduction

A business process (BP) consists of a set of activities which are performed in coordination in an organizational and technical environment (Weske, 2007) and which jointly realize a business goal. In the last years, there exists a growing interest in aligning information systems in a process-oriented way (Weske, 2007; Dumas et al., 2005; La Rosa et al., 2011; Huang et al., 2010).

In such context, BP management (BPM) can be seen as supporting BPs using methods, techniques, and software in order to design, enact, control, and analyze operational processes involving humans, organizations, applications, and other sources of information. Typically, the traditional BPM life cycle (Weske, 2007) includes several phases: process design & analysis, system configuration, process enactment and evaluation. The BP design & analysis phase has the goal to generate a BP model, i.e., to define the set of activities and the execution constraints between them (Weske, 2007) by formalizing the informal BP description using a particular BP modelling notation. After that, in the system configuration phase, BP models are implemented by configuring a Process-Aware Information System (PAIS). In the process enactment phase, in turn, BP instances are executed as defined in the BP model. Lastly, in the evaluation phase, information regarding the BP enactment is evaluated in order to identify and improve the quality of the BP model and their implementations. For that, enactment logs are traditionally analyzed by using BP activity monitoring and process mining techniques (van der Aalst, 2011; Liao et al., 2012). Process mining includes automated process discovery, conformance checking (i.e., monitoring deviations by comparing model and log), social network/organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations (van der Aalst et al., 2011).

1.1. Motivation

Nowadays, the accurate management of BPs is receiving increasing attention. In such a setting, conformance checking (Rozinat and van der Aalst, 2008) (which consists of verifying whether the observed behavior recorded in an event log during the process enactment matches the modelled behavior in the process design & analysis phase) is critical in many domains, e.g., process auditing

* Corresponding author. Tel.: +34 954 556 234; fax: +34 954 557 139.
E-mail addresses: dianabn@us.es (D. Borrego), irenebr@us.es (I. Barba).

http://dx.doi.org/10.1016/j.eswa.2014.03.010
0957-4174/

or risk analysis (De Leoni et al., 2012). This way, conformance checking is framed in the evaluation phase of the BPM life cycle (Weske, 2007) as one of the main application scenarios of process mining (van der Aalst, 2011; van der Aalst et al., 2011).

Furthermore, the economic success of an enterprise increasingly depends on its ability to react to changes in a quick and flexible way. Therefore, flexible PAISs (Reichert and Weber, 2012) are required to allow companies to rapidly adjust their BPs to changes. Typically, processes are specified in an imperative way. However, declarative process models, which implicitly specify the allowed behavior with rules that must be followed during the BP execution, are increasingly used. In a declarative model (e.g., constraint-based model) everything that is not forbidden is allowed. Therefore, the declarative specification of processes is an important step towards the flexible management of PAISs (van der Aalst et al., 2009).

Declarative BP modelling languages based on LTL (Linear Temporal Logic), e.g., Declare (Pesic, 2008), can be fruitfully applied in the context of compliance checking (Burattin et al., 2012). However, whereas there exist solid conformance checking techniques for imperative models (e.g., Rozinat and van der Aalst, 2008; Adriansyah et al., 2011; de Leoni and van der Aalst, 2013), little work has been conducted for declarative models (De Leoni et al., 2012).

In existing proposals in the context of conformance checking, only the control-flow perspective of BPs is typically considered. In fact, the data-flow of a process is usually only considered for determining its control-flow (e.g., which option of a XOR gateway is selected for execution). However, other perspectives (e.g., data) are also crucial when modelling and executing a process. In this respect, we propose to go one step further by incorporating to the process specification business data rules to describe the data semantics for the representation of the relations between data values in a process model. Compared to traditional proposals, the fact of including business data rules in a process allows to decrease the cost related to the modification of its business logic, to shorten the development time, to externalize and easily share rules among multiple applications, and to make changes faster and with less risk (Weber et al., 2009).

Moreover, in the context of conformance checking, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics (De Leoni et al., 2012; Burattin et al., 2012).

Summarising, the main research motivations of the proposed approach are: (1) the conformance checking of BPs is receiving increasing attention and is becoming critical in many domains, (2) the declarative specification of processes is an important step towards the flexible management of PAISs, which is required to allow companies to rapidly adjust their BPs to changes, (3) whereas there exist solid conformance checking techniques for imperative models, little work has been conducted for declarative models, (4) in existing proposals in the context of conformance checking, only the control-flow perspective of BPs is typically considered, although other perspectives (e.g., data) are also crucial when modelling and executing a process, and (5) in the context of conformance checking, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics.

1.2. Contribution

In order to enhance the accurate management of flexible BPs, this work presents an approach for performing conformance checking and for providing related diagnostics over declarative BP models which include business data rules. With this aim, efficient constraint-based approaches are proposed to allow for an increased effectiveness when performing conformance checking as well as when dealing with the diagnosis. Specifically, for the conformance checking process, one constraint-based approach which includes the modelling of the declarative process models as constraint satisfaction problems (CSPs) as well as a set of global constraints implemented through filtering rules is proposed. For the diagnosis process, such CSPs are transformed to Max-CSPs (i.e., CSPs which allow the violation of some constraints) to identify the minimum set of constraints of the declarative models which need to be relaxed/ removed to make such model feasible according to given process instances. For solving such Max-CSPs two different techniques (i.e., (1) using reified constraints and (2) attaining the Minimal Unsatisfiable Subsets) are considered.

Moreover, to check the effectiveness and efficiency of the proposed approach, the analysis of different performance measures over a diversified set of correct and representative models entailing different complexities has been performed. Results demonstrate the effectiveness of the proposed approach as well as show that the execution times which are obtained for both conformance checking and diagnosis are rather low for the considered models.

Note that the proposed approach is focused on addressing realistic problems from different domains which present flexible nature and where the accurate management of BPs is crucial (e.g., medical guidelines, processes from automotive industry and flight planning (Lanz et al., 2012)).

There is some related work on conformance checking over declarative BP models (e.g., Burattin et al., 2012; De Leoni et al., 2012; Grando et al., 2012; Montali et al., 2010; Grando et al., 2013; de Leoni et al., 2014). However, the approach presented in this paper differs from existing approaches in various ways: (1) besides the control-flow constraints, our approach is able to deal with business data rules; and (2) the proposed approach is based on constraint programming, which: (a) allows for efficient conformance checking and diagnosis processes, and (b) due to its versatility, facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities).

Summarising, the strengths of the proposed approach are listed as follows:

- it allows the BPs to be specified in a declarative way, which is an important step towards the flexible management of PAISs (van der Aalst et al., 2009),
- since the considered declarative language is based on high-level constraints, the problems can be modelled in an easy way,
- unlike existing proposals, our approach allows including business data rules in the process specification,
- unlike the current approach, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics (De Leoni et al., 2012; Burattin et al., 2012),
- since the proposed approach is based on constraint programming, it allows for efficient conformance checking and diagnosis processes at the same time as facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities).

However, the proposed approach also presents some weaknesses which are listed as follows:

- the business analysts must deal with a non-standard language for the declarative specification of BPs, therefore a period of training is required to let the business analysts become familiar with Declare specifications,

- the considered constraint-based models deal with both control-flow and data perspectives, but do not consider the resource perspective,
- additional evaluations of our proposal in the context of real process executions are desired and are intended to be addressed as future work.

The rest of the paper is organized as follows: Section 2 gives backgrounds on related areas, Section 3 presents the proposed approach, Section 4 deals with the evaluation, Section 5 presents a critical discussion of our proposal, Section 6 summarizes related work, and Section 7 concludes the paper. To improve the readability, the set of acronyms which appear throughout the contribution are included in Section 8.

2. Background

In this work, the declarative language Declare[1] Pesic (2008) is used as basis for specifying constraint-based process models.

Definition1. A **constraint-based process model** $S = (A, C_{BP})$ consists of a set of activities A and a set of constraints C_{BP} limiting execution behaviors.

The activities of a constraint-based process model can be executed arbitrarily often if not restricted by any constraints. In a Declare model control-flow constraints are specified trough templates, which are grouped into:

1. Existence templates: unary relations stating the number of times one activity is executed, e.g., Exactly (N,A) specifies that A should be executed exactly N times.
2. Relation templates: positive binary relations used to establish what should be executed, e.g., Precedence (A,B) specifies that before any execution of B at least one execution of A must be executed.
3. Negation templates: negative relations used to forbid the execution of activities in specific situations, e.g., NotCoexistence (A,B) specifies that if B is executed, then A cannot be executed, and vice versa.

In addition to control-flow relations, we consider business data constraints to describe the data semantics for the representation of relations between data values in a process model, resulting in a data constraint-based process model.[2] These constraints can be used for describing both the behavior of each single activity independently and the behavior of the overall process in a global way.

Definition 2. A **data constraint-based process model** $SData = (A, Data, C_{BP}, C_{Data})$ is a constraint-based model $S = (A, C_{BP})$ (cf. Definition 1) which also includes a set of data variables *Data*, and a set of business data constraints C_{Data} relating the variables included in *Data*.

Therefore, a data constraint-based process model includes control-flow and dataflow perspectives of a business process. The approach in Borrego et al. (2013) also presents a computational model for representing both perspectives, called workflow data graphs, which models topology and semantics for an imperative business process model, including business data constraints to express the behavior of the activities composing the model.

Business data constraints are linear or polynomial equations or inequations over data-flow variables, related by a boolean combination, according to a Backus Normal Form (BNF) grammar.

Definition 3. Let $v \in Data$ be a variable, *int_val* be an integer value, *nat_val* be a natural value, and *float_val* be a float value. The set of business data constraints which can be included in C_{Data} is generated by the following grammar in BNF:

$$
\begin{aligned}
&Constraint ::= Atomic\_ConstraintBOOL\_OPConstraint \\
&\quad |Atomic\_Constraint|\neg Constraint|(Constraint) \\
&BOOL\_OP ::= \wedge|\vee|\rightarrow \\
&Atomic\_Constraint ::= Function\ PREDICATE\ Function \\
&Function ::= v\ FUNCTION\ Function \\
&\quad |v|int\_val|nat\_val|float\_val \\
&PREDICATE ::= <|\leqslant|=|>|\geqslant \\
&FUNCTION ::= +|-|*
\end{aligned}
$$

Fig. 1 shows a data constraint-based process model which represents a hypothetical travel agency.[3] This agency manages holiday bookings by offering: transport (*TS*), accommodation (*AS*), and guided excursions (*GE*). After the client request (*G*) is carried out, the agency must write a report (*WR*) containing the information in answer to the request, which will then be sent to the client. Likewise, the example includes business data rules expressing the behavior of the elements in the control-flow perspective, like the business data constraint $WR.total\_cost \leqslant G.cost\_limit$ to indicate that the total cost of the holiday cannot surpass the cost limit. All these business data rules are defined in accordance to the BNF grammar in Definition 3, which provides a higher expressiveness compared to existing approaches.

This process is specified as the following data constraint-based process model (cf. Definition 2): $SData = (\{G, AS, GE, TS, WR\}, \{G.cost\_limit, AS.class, AS.cost, GE.cost, TS.class, TS.cost, WR.total\_cost\}, \{Exactly(1,G), Succ(G,AS), Succ(G,GE), Succ(G,TS), Prec(AS,GE), Prec(TS,GE), Resp(AS,WR), Resp(GE,WR), Resp(TS,WR), Prec(G, WR)\}, \{AS.class - TS.class \geqslant -1, AS.class - TS.class \leqslant 1, WR.total\_cost = AS.cost + GE.cost + TS.cost, WR.total\_cost \leqslant G.cost\_limit\})$. Note that in the graphical notation, business data constraints are depicted linked to activities and relations, as proposed by Montali (2009).

As the execution of a data constraint-based model proceeds, related information is recorded in an execution trace.

Definition 4. Let $SData = (A, Data, C_{BP}, C_{Data})$ be a data constraint-based process model (cf. Definition 2). Then: a **trace** $\sigma = (Values, E)$ is composed of: (1) $Values = \{(Data_i, Value_i), Data_i \in Data\}$, which is a set of pairs $(Data_i, Value_i)$ stating the $Value_i$ assigned to $Data_i$ in the trace; and (2) E, which is a sequence of starting and completing events $< e_1, e_2, \ldots e_n >$ regarding activity executions a_i, $a \in A$, i.e., events can be:

1. $start(a_i, T)$, meaning that the i-th execution of activity a is started at time T.
2. $comp(a_i, T)$, meaning that the i-th execution of activity a is completed at time T.

A process instance represents a concrete execution of a data constraint-based process model and its execution state is reflected by the execution trace.

Definition 5. Let $SData = (A, Data, C_{BP}, C_{Data})$ be a data constraint-based process model. Then: a **process instance** $I = (SData, \sigma)$ is defined by *SData* and a corresponding trace σ.

A running process instance $I = (SData, \sigma)$ is **compliant** with $SData = (A, Data, C_{BP}, C_{Data})$ if σ satisfies all constraints stated in

[1] Declare is one of the most referenced and used declarative BP languages in the context of BPM.

[2] In a similar way, (Montali, 2009) deals with modelling data in Declare.

[3] This model is an adaptation of the model presented in Barba et al. (2013b).

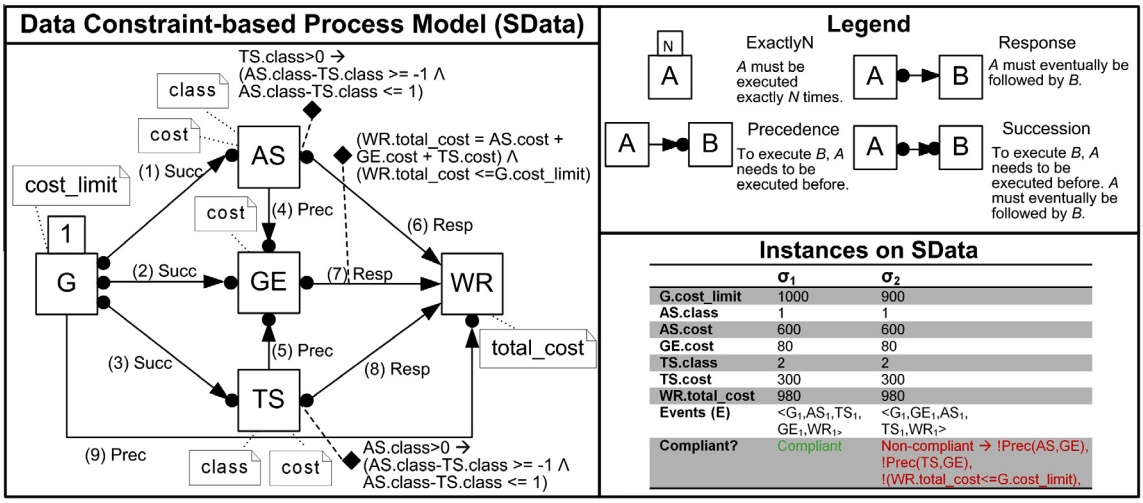

Fig. 1. Running example: a data constraint-based process model and some related instances.

$C_{BP} \cup C_{Data}$. In this way, when checking the conformance of a process instance, both the control-flow and the data-flow conformances are checked. As examples, Fig. 1 includes one compliant instance and one non-compliant instance[4] due to the constraints $\{Prec(AS, GE),\ Prec(TS, GE),\ WR.total\_cost <= G.cost\_limit\}$ are not satisfied.

There are few proposals for checking the conformance of constraint-based process models (e.g., Grando et al., 2012; Burattin et al., 2012; De Leoni et al., 2012). However, we are not aware of any constraint-based approach (i.e., based on constraint programming) for checking the conformance in constraint-based process models. To solve a problem through constraint programming (CP) (Rossi et al., 2006), it needs to be modelled as a constraint satisfaction problem (CSP).

Definition 6. A **CSP** $P = (V,\ D,\ C_{CSP})$ is composed of a set of variables V, a domain of values D for each variable in V, and a set of constraints C_{CSP} between variables, so that each constraint represents a relation between a subset of variables and specifies the allowed combinations of values for these variables.

A solution for a CSP consists of assigning values to all the CSP variables. A solution is feasible when the assignment of values to variables satisfies all the constraints. In a similar way, a CSP is feasible if there exists at least one related feasible solution, overconstrained otherwise. To solve the latter, Max-CSPs (which are CSPs which allow the violation of some constraints) can be considered within two different techniques:

1. Use of reified constraints, which are constraints associated to CSP boolean variables which denote its truth value. Specifically, the identification of the constraints to relax (or remove) to make the overconstrained model feasible can be performed by maximizing the number of reified constraints whose truth value is equal to true. This gives rise to a constraint optimization problem (COP).

Definition 7. A **COP** $P_{OF} = (V,\ D,\ C_{CSP},\ OF)$ is a CSP which also includes an objective function OF to be optimized.

2. Attainment of the Minimal Unsatisfiable Subsets (MUSes), which represent the most succinct explanations, in terms of the number of involved constraints, of infeasibility, according to the next definition.

Definition 8. Given a set of constraints C within a CSP, a Minimal Unsatisfiable Subset (*MUS*) of C is a subset of C that is (1) unsatisfiable and (2) minimal, in the sense that removing any one of its elements makes the rest of the *MUS* satisfiable. That is, m is a *MUS* of C iif $m \subseteq C : m$ is unsatisfiable, and $\forall c \in m,\ m \setminus \{c\}$ is satisfiable.

Constraint programming allows the separation of the models from the algorithms, so that once a problem is modeled as a CSP or a COP, a generic or specialized constraint-based solver can be used to obtain the required solution. Several mechanisms are available for the solution of CSPs and COPs (e.g., complete search algorithms, which perform a complete exploration of a search space which is based on all possible combinations of assignments of values to the CSP variables). Regardless of the search which is used, global constraints (constraints capturing a relation between a non-fixed number of variables) can be defined to improve the modelling of the problems. The global constraints can be implemented through filtering rules (rules responsible for removing values which do not belong to any solution) to efficiently handle the constraints.

3. Our proposal

Although a declarative process model is verified correctly from the control-flow and data-flow perspectives at build-time, the process may not behave correctly at run-time. This anomaly can be detected by performing conformance checking by means of the comparison between the instances collected in the event log, which reflect the actual behavior of the process, and the declarative model indicating the expected behavior. The discrepancies detected after the execution of process instances can be derived from: (i) a declarative model that has not been correctly designed in accordance with the business goals; or (ii) an abnormal execution of the process.

In this work, a method for checking the conformance of declarative models which include business data constraints and for diagnosing the parts of the model that were not executed as they were modelled is proposed (cf. Fig. 2). The method starts from a data constraint-based process model (cf. Definition 2) defined by a business analyst. During the process enactment, the events related to the start and the completion of activity executions and to data are

[4] For the sake of clarity, only completed events for activity executions are included in the trace representation.

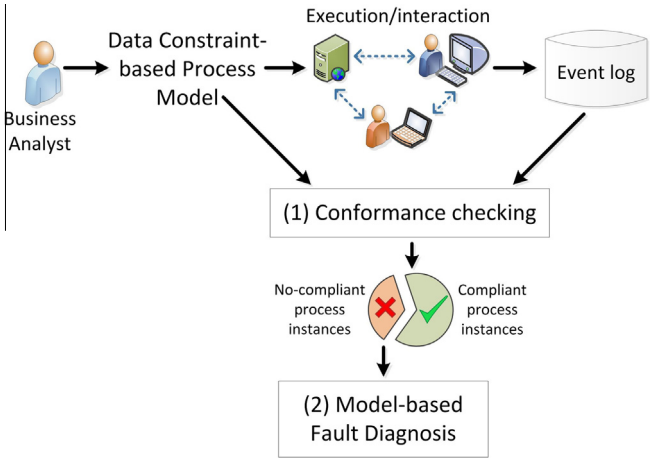

Fig. 2. Conformance checking and diagnosis for data constraint-based process models.

stored in an event log, collecting the process instances (cf. Definition 5). To detect discrepancies, conformance checking is performed over the input model and the instances recorded in the event log (cf. Section 3.1). As a result, the instances are classified as *compliant* or *non-compliant* if they comply or fail to comply with the model, respectively. Finally, using the *non-compliant* traces, the model-based fault diagnosis of the process model is performed (cf. Section 3.2). This diagnosis determines the parts of the model that were either incorrectly designed or abnormally executed by users.

3.1. Conformance checking

To check the conformance of a process instance, a constraint-based approach is proposed. To deal with the fact that in Declare BP activities can be executed arbitrarily often if not restricted by any constraint, in the proposed constraint-based approach process activities are modelled as repeatable activities (cf. Definition 9).

Definition 9. A **repeatable activity** $ra = (nt,\ a,\ sActs)$ is an activity a which can be executed several times, where nt is a CSP variable specifying the number of times the activity is executed (which is the number of activity occurrences related to ra), and $sActs$ is a sequence of its related activity occurrences ($sActs = [a_1, \ldots, a_{nt}]$).

For the sake of brevity we write $nt(ra)$, $sActs(ra)$, etc., when referring to attributes nt, $sActs$, etc., of the repeatable activity ra. As an example, for the data constraint-based model of Fig. 1 there are 5 repeatable activities (e.g., $(nt(G),\ G, [G_1, \ldots, G_{nt(G)}])$).

For each repeatable activity, nt activity occurrences exist (cf. Definition 10).

Definition 10. An **activity occurrence** $a_i = (st, et, ra)$ represents the i-th execution of a repeatable activity ra ($sActs(ra)[i] = a_i$), where st and et are CSP variables which refer to the start and the end times of the activity occurrence, respectively.[5]

As an example, related to the activity G of the data constraint-based model of Fig. 1 there are $nt(G)$ activity occurrences (e.g., $(st(G_1),\ et(G_1), G)$).

Furthermore, to improve the modelling of the problems and to efficiently handle the constraints when checking the satisfiability of a process instance, the proposed constraint-based approach includes a global constraint implemented through a filtering rule for each Declare template. For a detailed description of all the filtering rules which have been developed for such purpose see (Barba and Del Valle, 2011).

Moreover, to model the problem as a CSP, a CSP-Conformance problem (cf. Definition 11) related to a data constraint-based process model is created.

Definition 11. Let $SData = (A,\ Data,\ C_{BP},\ C_{Data})$ be a data constraint-based process model (cf. Definition 2), and $RActs$ be the set of repeatable activities related to $SData$ ($RActs = \{ra = (nt,\ a,\ sActs),\ a \in A\}$. Then: a **CSP-Conformance problem** related to $SData$ and $RActs$ is a CSP $P = (V,\ D,\ C_{CSP})$ (cf. Definition 6), where:

- The set of variables $V = \{nt(ra),\ ra \in RActs\} \cup \{st(a_i),\ et(a_i),\ a_i \in sActs(ra),\ ra \in RA\} \cup \{Data_i, Data_i \in Data\}$.
- The set of domains D is composed of the domains for each variable from V.
- The set of constraints C_{CSP} is composed of: (1) the global constraints (implemented by the filtering rules) related to the Declare constraints included in C_{BP} and (2) the constraints related to the business data constraints included in C_{Data}.

For the data constraint-based process model of Fig. 1:

- V = {$nt(G)$, $nt(AS)$, $nt(GE)$, $nt(TS)$, $nt(WR)$, $st(G_1)$, $et(G_1)$, …, $st(WR_{nt(WR)})$, $et(WR_{nt(WR)})$, G_cost_limit, AS_class, AS_cost, GE_cost, TS_class, TS_cost, WR_total_cost},
- D is composed of the domains of each variable in V, and
- C_{CSP} = {$Exactly(1, G)$, $Succ(G, AS)$, $Succ(G, GE)$, $Succ(G, TS)$, $Prec(AS, GE)$, $Prec(TS, GE)$, $Resp(AS, WR)$, $Resp(GE, WR)$, $Resp(TS, WR)$, $Prec(G, WR)$, $|AS\_class - TS\_class| \leqslant 1$, $WR\_total\_cost = AS\_cost + GE\_cost + TS\_cost$, $WR\_total\_cost \leqslant G\_cost\_limit$}.

Let $I = (SData, \sigma)$ be a process instance (cf. Definition 5) defined by the data constraint-based process model $SData = (A,\ Data,\ C_{BP},\ C_{Data})$ (cf. Definition 2) and the trace $\sigma = (Values,\ E)$ (cf. Definition 4). Moreover, let $P = (V,\ D,\ C_{CSP})$ be the CSP-Conformance problem related to $SData$ (cf. Definition 11). Then, the problem of checking the conformance of I regarding $SData$ is equivalent to checking the feasibility of P when instantiating its CSP variables as follows:

- $\forall a \in A,\ nt(a) = argmax_i(comp(a_i, T) \in E)$, stating that the number of times a repeatable activity a is executed is instantiated to the index of the last execution of a which was completed in the trace,
- $\forall a \in A,\ i \in [1..nt(a)], st(a_i) = T\ |(start(a_i, T) \in E)$, stating that the start time of each activity occurrence is instantiated to the time when this occurrence started in the trace,
- $\forall a \in A,\ i \in [1..nt(a)], et(a_i) = T\ |(comp(a_i, T) \in E)$, stating that the end time of each activity occurrence is instantiated to the time when this occurrence was completed in the trace,
- $\forall Data_i \in Data,\ Data_i = Value_i\ |(Data_i, Value_i) \in Values$, stating that all the variables related to data are instantiated with their actual values in the trace.

In this way, the process instance I is compliant with $SData$ if the CSP-Conformance problem P instantiated as mentioned is feasible, non-compliant otherwise.

[5] Note that the CSP variables st and et allows dealing with non-atomic activities.

3.2. Model-based fault diagnosis process

For each non-compliant instance, a model-based fault diagnosis process (de Kleer and Kurien, 2003) is applied to determine which parts of the model are potentially the source of the problem.

The proposed diagnosis process has the goal of identifying the minimum set of constraints of the data constraint-based process model (cf. Definition 2) which need to be relaxed/removed to make such model feasible according to a given instance I (which is equivalent to solve the related Max-CSP, cf. Section 2). To this end, the two aforementioned techniques to solve Max-CSPs are used as detailed in the following.

3.2.1. Diagnosis using reified constraints

This proposal builds reified constraints based on the constraints which are included in a CSP-Conformance problem. A reified constraint consists of a constraint and a variable which denotes its truth value. This way, starting from a CSP composed of a set of constraints, it is necessary to add new variables to this CSP. They are boolean variables, one for each constraint, which are used to build reified constraints, associating each constraint to a boolean through an equality.

To this end, the related CSP-Conformance problem $P = (V, D, C_{CSP})$ (cf. Definition 11) is modified (by including reified constraints) and translated into a COP $P_{OF} = (V', D', C'_{CSP}, OF)$ (Borrego et al., 2010) where:

- $V' = V \cup RefVars \cup OF$, where $RefVars$ is the set of the boolean CSP variables which hold the truth values of the reified constraints (note that there is the same number of boolean variables in $RefVars$ as the number of reified constraints),
- $D' = D \cup Dom(RefVars) \cup Dom(OF)$, where $Dom\ (RefVars)$ is the set of domains of each variable in $RefVars$,
- All the constraints included in C_{CSP} are transformed into reified constraints, in such a way that C'_{CSP} is composed of: (1) the reified constraints related to the global constraints which are included in C_{BP} and (2) the reified constraints related to the business data constraints which are included in C_{Data} (cf. Definition 11).
- $OF = max|TrueRefVars|$, where $TrueRefVars = \{v ar \in RefVars, v ar = true\}$.

The fact of transforming a constraint into a reified constraint makes the satisfaction of such constraint optional since a solution to a CSP which violates any reified constraints is a feasible solution. Most of the variables of this COP are instantiated as they were when checking the conformance of I (i.e., all the CSP variables except the reified variables and OF are instantiated according to I). However, after transforming the constraints into reified constraints, the instantiation of the CSP-Conformance problem according to I becomes feasible. Moreover, since the goal consists of diagnosing the source of the problem, the maximization of the number of reified constraints whose boolean value is equal to true is included as the objective function to be optimized.

Solving COPs typically takes exponential time due to the dependency of their complexity on the number of values each variable can take (Kumar, 1992). The proposed constraint-based approach is based on the first-fail principle (Van Hentenryck, 1989), which orders the variables by increasing set cardinality, breaking ties by choosing the variable with the smallest domain size, and reducing the average depth of branches in the search tree. Therefore, as shown in Section 4, the time it takes to check the conformance of declarative models with the presented approach is acceptable.

After solving this COP, the minimum set of constraints to be relaxed/removed to obtain a model compliant with the instance I is equal to the constraints related to reified variables which are false in the solution to the COP.

As an example, for the CSP-Conformance problem of the model in Fig. 1, V is extended, including $\{exactlyG, succG\_AS, succG\_GE, succG\_TS, precAS\_GE, precTS\_GE, resp\ AS\_WR, respGE\_WR, respTS\_WR, precG\_WR, dc1, dc2, dc3, OF\}$. D is also extended with the domain of the new variables, and C_{CSP} is modified by transforming each constraint $c \in CSP$ into $rVarC == (c)$, being $rVarC$ the reified variable related to c (e.g., $exactlyG == (Exactly(1, G))$ and $dc1 == (|AS\_class - TS\_class| \leqslant 1)$), and by including the constraint related to OF.

3.2.2. Diagnosis through the determination of MUSes

This solution is based on the attainment of all the MUSes, which represent the most succinct explanations, in terms of the number of involved constraints, of infeasibility. Indeed, when we check the consistency of a CSP, it is preferable to know which constraints are contradicting one another rather than only knowing that the CSP as a whole is inconsistent.

Due to the computational complexity of the attainment of all MUSes, some existing approaches were designed to derive only some of the MUSes and not all of the MUSes of an overconstrained CSP. Our proposal is based on an improvement in Gasca et al. (2007), which is grounded in structural analysis in order to determine all the MUSes efficiently. In that paper, several techniques are presented, which improve the complete technique in several ways depending on the structure of the constraint network. These techniques make use of the powerful concept of the structural lattice of the constraints and neighbourhood-based structural analysis to boost the efficiency of the exhaustive algorithms. Since systematic methods for solving hard combinatorial problems are too computational expensive, structural analysis offers an alternative approach for fast generation of all MUSes. Accordingly, experimental studies of these new techniques outperform the best exhaustive techniques since they avoid the necessity of solving a high number of CSPs with exponential complexity. However they do add certain new procedures with polynomial complexity.

In the case of this contribution, two techniques are applied:

- **Exhaustive technique.** It attains the *MUSes* in accordance with the process described in Algorithm of Fig. 3. It uses the queue Q (line 3) to initially store the inconsistent constraints. The process uses this queue to exhaustively check the satisfiability of every possible combination of constraints. In the case a subset of constraints is proved unsatisfiable (checked in line 10), it is stored in the list *MUS* (line 13).
- **Variable-Based Neighbourhood technique.** It uses the knowledge about the Variable-Based Neighbourhood explained in Gasca et al. (2007). To this end, the constraints in the CSP-Conformance problem $P = (V, D, C_{CSP})$ (cf. Definition 11) are related between them as neighbours, so that two constraints c_1 and c_2 are neighbours if they share variables (i.e. at least one variable v_i appears in both c_1 and c_2).

Algorithm of Fig. 4 shows the details of this process, with is similar to the process described in Algorithm of Fig. 3, but with the difference of the generation of neighbours to propagate the search of MUSes (line 8).

As an improvement upon the technique presented in Gasca et al. (2007) when it comes to calculate the neighbours of a set of business data constraints, not all the neighbours are generated: an order is established between the constraints, so that only those neighbours that are subsequent to all the constraints in the set are generated. In this way we succeed in generating only *new* neighbours, thereby avoiding the repeated study of the satisfiability of the same collection of constraints.

```
1:  procedure ObtainMUSes1(Constraints C)
2:      //Being C the constraints that present the inconsistency:
3:      Q := Queue initialized with each constraint in C
4:      MUS := List that will contain the MUSes, initialized to empty
5:      while Q is not empty do
6:          {c_i ... c_j} := Q.poll() //retry and remove the head of Q
7:          for c_k ∈ {c_{j+1} ... c_n} do
8:              if NOT ∃SubSet^{1...n-1}_{{c_i...c_j}} ∪ c_k ∈ MUS then
9:                  // being n the cardinality of {c_i ... c_j}
10:                 if {c_i ... c_j} ∪ c_k is a satisfiable CSP then
11:                     Q.add({c_i ... c_j} ∪ c_k)
12:                 else
13:                     MUS.add({c_i ... c_j} ∪ c_k)
14:                 end if
15:             end if
16:         end for
17:     end while
18: end procedure
```

Fig. 3. Algorithm to obtain the *MUSes* (I).

Once all *MUSes* have been obtained, it is necessary to obtain the minimal set of constraints that make the whole CSP unsatisfiable. This way, it is possible to identify the minimum set of constraints of the data constraint-based process model (cf. Definition 2) which need to be relaxed/removed to make such model feasible according to a given instance. To this end, the calculation of the minimal hitting sets (cf. Definitions 12 and 13) is performed to find out those constraints.

Definition 12. Hitting Set (HS) for a collection of sets of components C is a set of components $H \subseteq \bigcup_{S \in C} S$ such that H contains at least one element for each $S \in C$.

Definition 13. A HS of C is minimal iff no proper subset of H is an HS of C. The minimal HSs for a set of sets are formed by $\{H_1, \ldots, H_n\}$, where H_i is a minimal HS of components.

The calculation of these minimal hitting sets of constraints provides us with the minimal diagnosis for a CSP-Conformance problem, since it obtains the minimal sets of constraints, representing parts of the model, which are responsible for the no conformance.

4. Case study

The purpose of this section consists of analysing the effectiveness and efficiency of the proposed approach (detailed in Section 3) over a diversified set of data constraint-based process models (cf. Definition 2). Specifically, this section provides an empirical study for: (1) checking the conformance of a set of process instances according to a given data constraint-based process model through the proposed approach (detailed in Section 3.1) as well as (2) performing a model-based fault diagnosis process for each non-compliant instance to determine which parts of the model are potentially the source of the problem through the proposed approach (detailed in Section 3.2). For such purpose, the case study protocol detailed in Brereton et al. (2008) is followed to improve the rigour and validity of this study. This way, this section is organized as follows: background (cf. Section 4.1), design (cf. Section 4.2), case selection (cf. Section 4.3), data collection (cf. Section 4.4), analysis (cf. Section 4.5), and plan validity (cf. Section 4.6).

```
1:  procedure ObtainMUSes2(Constraints C)
2:      //Being C the constraints that present the inconsistency:
3:      Q := Queue initialized with each constraint in C.
4:      //The data structure selected determines the search strategy.
5:      MUS := List that will contain the MUSes, initialized to empty
6:      while Q is not empty do
7:          {c_i ... c_j} := Q.poll() //choose an element of Q
8:          neighbours := expand({c_i ... c_j})
9:          //generate neighbours according to the Variable-Based Neighbourhood
10:         for all c_k ∈ neighbours do
11:             if {c_i ... c_j} ∪ c_k is a satisfiable CSP then
12:                 Q.add({c_i ... c_j} ∪ c_k)
13:             else
14:                 MUS.add({c_i ... c_j} ∪ c_k)
15:             end if
16:         end for
17:     end while
18: end procedure
```

Fig. 4. Algorithm to obtain the *MUSes* (II).

4.1. Background

As mentioned in Section 1 and detailed in Section 6, the approach presented in this paper differs from existing approaches related to conformance checking and diagnosis over declarative BP models (e.g., Burattin et al., 2012; De Leoni et al., 2012; Grando et al., 2012; Montali et al., 2010) in various ways: (1) besides the control-flow constraints, it is able to deal with business data rules; and (2) it is based on constraint programming, which allows for efficient conformance checking and diagnosis processes as well as facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities).

Taking such aspects into account, 2 main research questions (MRQs) related to the proposed approach (cf. Section 3) are addressed in the current case study:

- MRQ1, which checks the suitability of the proposed approach for carrying out the conformance checking of process instances according to a given data constraint-based process model (cf. Section 3.1), and
- MRQ2, which checks the suitability of the proposed approach for carrying out the model-based fault diagnosis process by either using reified constraints or through the determination of MUSes (cf. Section 3.2).

4.2. Design

For carrying out the experiments, a multiple-case design (i.e., over multiple data constraint-based process models, cf. Definition 2) is used since solving the aforementioned research questions require the use of problems of different complexity. Moreover, an embedded design is carried out since 3 analysis units (i.e., the conformance checking method proposed in Section 3.1, the diagnosis process through reified constraints proposed in Section 3.2.1, and the diagnosis process through MUSes proposed in Section 3.2.2) are individually studied. Therefore, the objects of study are: (1) the new conformance checking method (cf. Section 3.1), (2) the new model-based fault diagnosis process which uses reified constraints (cf. Section 3.2.1), and (3) the new model-based fault diagnosis process which is based on the determination of MUSes (cf. Section 3.2.2).

For answering the aforementioned research questions, additional subquestions arise. Specifically, 2 subquestions are proposed to address MRQ1:

- MRQ1(1), which checks the effectiveness of the proposed conformance checking method (i.e., does the proposed conformance checking method correctly verify whether a given process instance matches a given data constraint-based process model?), and

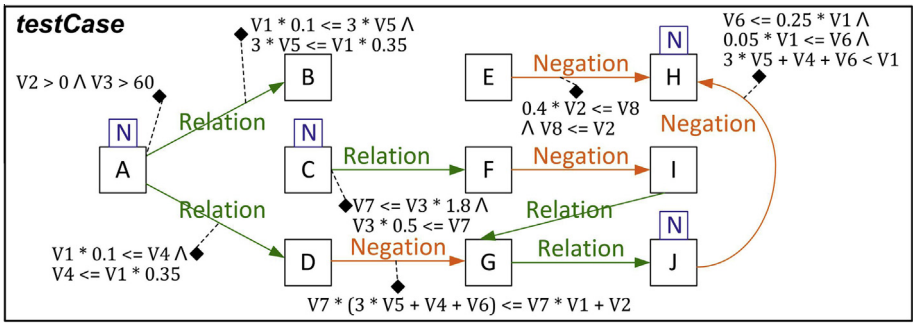

Fig. 5. Generic data constraint-based process model.

- MRQ1(2), which checks the efficiency of the proposed conformance checking method when solving problems of different complexity.

In a similar way, 5 subquestions are proposed to address MRQ2:

- MRQ2(1), which checks the effectiveness of the proposed diagnosis process by using reified constraints (i.e., does this process correctly identify the minimum set of constraints of the data constraint-based model which need to be relaxed/removed to make such model feasible according to a given instance?),
- MRQ2(2), which checks the efficiency of the proposed diagnosis process by using reified constraints when solving problems of different complexity,
- MRQ2(3), which checks the effectiveness of the proposed diagnosis process through the determination of MUSes (i.e., does this process correctly identify the minimum set of constraints of the data constraint-based model which need to be relaxed/removed to make such model feasible according to a given instance?),
- MRQ2(4), which checks the efficiency of the proposed diagnosis process through the determination of MUSes when solving problems of different complexity, and
- MRQ2(5), which checks whether any of the two proposed diagnosis methods is the most efficient in general regardless of the characteristics of the considered problems.

The measures to be used to investigate such subquestions are:

- *%Correct*: percentage of instances for which the proposed conformance checking method behaved correctly (related to MRQ1(1)),
- *checkTime*: average time (in milliseconds) for checking the conformance through the proposed approach (related to MRQ1(2)),
- OF_R: average value which is reached for the objective function to be minimized (which is the number of constraints which should be relaxed/removed to make the model compliant) when using reified constraints (related to MRQ2(1)),
- $diagTime_R$: average time (in milliseconds) for diagnosing when using reified constraints (related to MRQ2(2) and MRQ2(5)),
- OF_M: average value which is reached for the objective function to be minimized (which is the number of constraints which should be relaxed/removed to make the model compliant) when basing on the determination of MUSes (related to MRQ2(3)),
- $diagTime_M$: average time (in milliseconds) for diagnosing when basing on the determination of MUSes (related to MRQ2(4) and MRQ2(5)).

Note that the variables OF_R, $diagTime_R$, OF_M, and $diagTime_M$ only make sense when diagnosing is required (that is for non-compliant instances).

For checking the effectiveness of the proposed methods (i.e., answering subquestions MRQ1(1), MRQ2(1) and MRQ2(3)), it is necessary to know the correct solution for each case which is analyzed.

On the one hand, when checking the conformance, whether a given process instance matches a given data constraint-based process model has been verified through another constraint-based approach. Such approach considers the same variables when defining the CSP-Conformance problem than the one presented in Section 3.1 but differs from it in which neither global constraints nor filtering rules are included. Instead of that, the equivalent local constraints for each Declare constraint are included. As an example, the equivalent local constraint for the Precedence (A,B) global constraint is: $nt(B) > 0 \rightarrow (nt(A) > 0) \wedge (et(A_1) \leqslant st(B_1))$. For a complete description of the local constraints which are equivalent to each Declare global constraint, the reader is referred to Barba and Del Valle (2011).

On the other hand, when performing the diagnosis, whether both proposals (i.e., reified constraints and MUSes) correctly identify the minimum set of constraints of the data constraint-based model which need to be relaxed/removed to make such model feasible according to a given instance is checked by comparing the values obtained for OF_R and OF_M. In the case that such values match, it is considered that both techniques behave correctly since it is highly improbable that they lead exactly to the same incorrect solution.

4.3. Case selection

For the empirical evaluation of the proposed approach, a diversified set of data constraint-based process models (cf. Definition 2) are generated considering some important characteristics to be considered when analysing the effectiveness and efficiency of the proposed approach (cf. Section 3): (i) correctness, i.e., the models must represent feasible problems without conflicts and without any dead activities (none of the traces that satisfies the model contains this activity), and (ii) representativeness, i.e., the models must represent problems which are similar to actual BPs. Consequently, we require the test models to be of medium-size and comprise all three types of Declare templates (existence, relation, and negation, cf. Section 2). Moreover, the considered models must include business data constraints (cf. Definition 3). Overall, one template model *testCase* which contains 10 repeatable activities (cf. Definition 9), 9 control-flow constraints, and 8 data-flow constraints is considered as basis (cf. Fig. 5). Those data-flow constraints are defined in accordance with the grammar in Definition 3, being hence more expressive data-flow constraints than the ones considered in other approaches. Note that although this model includes only 10 activities, the related process instances generally contain much more activity executions (i.e, activity occurrences, cf. Definition 10) depending on the constraints which

are included, since such 10 activities are repeatable activities (cf. Definition 9), as detailed later.

During the experiments the aforementioned generic model is specified by instantiating the generic relations (depicted by the labels *Relation* and *Negation* in Fig. 5) with concrete constraints. The developed filtering rules for the different Declare constraints (cf. Section 3.1) significantly vary in their computational complexity, as detailed in Barba and Del Valle (2011) (cf. Table 1 for the different complexity groups, where n is the number of repeatable activities of the process model). This way, for covering all the complexity groups, 4 models (testCaseA, testCaseB, testCaseC, testCaseD) are generated by substituting the labels *Relation* and *Negation* by: (A) Response and Negation Response, (B) Response and Negation Alternate Response, (C) Alternate Response and Negation Response, and (D) Alternate Response and Negation Alternate Response, respectively.

Moreover, in the case of Existence templates, a value for label N must be established ($N \in \{10, 20, 30\}$ is considered). Note that the value given to N highly influences the actual number of activity occurrences of the process instances, which can vary between 40 and 300 in the considered generic model. This is due to, considering the values given to N, activities A, C, H, and J in *testCase* (cf. Fig. 5) are executed at a minimum 10 times and at most 30 times while activities B, D, E, F, G, and I are, in principle, executed at a minimum 0 times and at most 30 times.

Overall, considering the values given to *Relation*, *Negation* and N, 12 different specific models (*Model* $\in$ {*testCaseA10, testCaseA20, testCaseA30, testCaseB10, testCaseB20, testCaseB30, testCaSection 10, testCaSection 20, testCaSection 30, testCaseD10, testCaseD20, testCaseD30*}) are generated.

Furthermore, since temporal and control-flow perspectives are considered, activity durations are relevant and need to be stated. This is carried out by randomly varying activity durations between 1 and 10 to achieve diversified results, resulting in 750 games of durations. This way, the proposed approach is tested over $12 \times 750 = 9000$ data constraint-based process models, whose activity occurrences vary between 40 and 300. This set of models, besides containing a high number of cases, is generated considering different complexity groups and different repetitions for the repeatable activities (as previously detailed), resulting in a rather diversified set.

Table 1
Type and complexity of the filtering rules related to the Declare templates.

Template	Type	Complexity
ExistenceN (A)	Existence	$\Theta(1)$
AbsenceN (A)	Existence	$\Theta(1)$
ExactlyN (A)	Existence	$\Theta(1)$
Responded Existence (A,B)	Relation	$O(n)$
CoExistence (A,B)	Relation	$O(n)$
Precedence (A,B)	Relation	$O(n)$
Response (A,B)	Relation	$O(n)$
Succession (A,B)	Relation	$O(n)$
Alternate Precedence (A,B)	Relation	$O(n \times nt^3)$
Alternate Response (A,B)	Relation	$O(n \times nt^3)$
Alternate Succession (A,B)	Relation	$O(n \times nt^3)$
Chain Precedence (A,B)	Relation	$O(n \times nt^3)$
Chain Response (A,B)	Relation	$O(n \times nt^3)$
Chain Succession (A,B)	Relation	$O(n \times nt^3)$
Responded Absence (A,B)	Negation	$O(n)$
Negation Response (A,B)	Negation	$O(n)$
Negation Alternate Precedence (A,B)	Negation	$O(n \times nt^2)$
Negation Alternate Response (A,B)	Negation	$O(n \times nt^2)$
Negation Alternate Succession (A,B)	Negation	$O(n \times nt^2)$
Negation Chain Succession (A,B)	Negation	$O(n \times nt^3)$

For each specific model, compliant and non-compliant instances are randomly generated to create a diversified synthetic log. As mentioned, related to a specific declarative model there may exist multiple executions which meet the constraints (i.e., several feasible execution plans). In previous approaches (cf. Barba et al., 2013a, 2013b), feasible optimized enactment plans from Declare specifications were generated. To this end, activity durations and resource availabilities were considered. In the current work, the proposal presented in Barba et al. (2013a, 2013b) is used for generating the flow of both compliant and non-compliant instances related to the 9000 different models. Since resource perspective is out of the scope of this paper, this aspect is not considered when generating the instances. The generation of both compliant and non-compliant instances is detailed as follows:

1. Generation of compliant instances: The approach presented in Barba et al. (2013a, 2013b) is used to generate an enactment plan related to each combination of specific model plus a specific sample of activity durations (9000 compliant instances are generated). Regarding data, 750 combinations of values which are compliant with the business data rules (cf. Fig. 5) are randomly generated by using a simple constraint-based approach, resulting in 750 compliant games of values for the data variables.
2. Generation of potentially non-compliant instances: To this end, the approach presented in Barba et al. (2013a, 2013b) is also used. Specifically, 9000 potentially non-compliant enactment plans are generated by randomly removing one of the Declare relations or negations of the considered models. Regarding data, 750 combinations of values which are not compliant with the business data rules (cf. Fig. 5) are randomly generated by using a simple constraint-based approach, resulting in 750 non-compliant games of values for the data variables.

From this, 4 groups of tests are generated, each one containing 9000 instances (i.e., 36000 instances are generated in total): (1) compliant instances; (2) instances non-compliant in data; (3) instances potentially non-compliant in control-flow; and (4) instances potentially non-compliant in control-flow and non-compliant in data. Note that the way in which the instances non-compliant in data are generated always ensures the non-compliance. However, the way in which the instances non-compliant in control-flow are generated may lead to compliance. In fact, the percentage of compliant instances in the complete set of generated instances is 38.9% (specifically 13998 instances out of 36000) instead of 25%. The complete set of process instances which are generated for the empirical evaluation can be accessed at http://quirce.lsi.us.es/thesis/EventLog.zip.

To solve the constraint-based problems, a complete search algorithm (cf. Section 2) based on the first-fail heuristic is integrated in the COMET^{TM} system (Dynadec, 2010). This algorithm is run on a Intel Core I7 processor, 3.4 GHz, 8 GB RAM, running Windows 7.

4.4. Data collection

For collecting the resulting data, the response variables (cf. Section 4.2) are calculated for both conformance checking (cf. Table 2) and diagnosis (cf. Table 3) processes by considering the average values for the instances of each model (*Model*) differentiating between compliant and non-compliant instances in the case of conformance checking. In these tables, besides the identifier of the model and the values for the response variables, each row also includes additional information: #*inst*, which is the number of compliant/non-compliant instances related to that model which are included in the complete set of instances (i.e., the number of instances used for calculating the averages) and #*actOcc*, which is

Table 2
Experimental results for conformance checking.

Tested model	*#Inst*	*#ActOcc*	%Correct	*CheckTime*
Compliant instances				
testCaseA10	1170	43	100	267.91
testCaseB10	1165	43	100	268.82
testCaseC10	1168	70	100	268.59
testCaseD10	1167	70	100	268.59
testCaseA20	1167	83	100	316.18
testCaseB20	1165	83	100	316.17
testCaseC20	1165	140	100	319.15
testCaseD20	1167	140	100	319.44
testCaseA30	1167	123	100	317.47
testCaseB30	1165	123	100	318.35
testCaseC30	1165	210	100	327.49
testCaseD30	1167	210	100	328.52
Non-compliant instances				
testCaseA10	1831	43	100	267.58
testCaseB10	1834	43	100	267.80
testCaseC10	1831	70	100	269.29
testCaseD10	1834	70	100	268.91
testCaseA20	1834	83	100	311.54
testCaseB20	1834	83	100	312.50
testCaseC20	1834	140	100	315.80
testCaseD20	1834	140	100	317.07
testCaseA30	1834	123	100	313.61
testCaseB30	1834	123	100	314.59
testCaseC30	1834	210	100	318.63
testCaseD30	1834	210	100	319.25

Table 3
Experimental results for diagnosing.

Tested model	*#Inst*	*#ActOcc*	$DiagTime_R$	OF_R	$DiagTime_M$	OF_M
Non-compliant instances						
testCaseA10	1831	43	272.61	2.49	24983	2.49
testCaseB10	1834	43	272.53	2.49	24501	2.49
testCaseC10	1831	70	272.39	2.49	24540	2.49
testCaseD10	1834	70	271.78	2.61	24607	2.61
testCaseA20	1834	83	318.98	2.49	26020	2.49
testCaseB20	1834	83	318.52	2.49	26260	2.49
testCaseC20	1834	140	321.00	2.59	26329	2.59
testCaseD20	1834	140	321.16	2.61	26425	2.61
testCaseA30	1834	123	319.79	2.49	28004	2.49
testCaseB30	1834	123	318.82	2.49	27969	2.49
testCaseC30	1834	210	323.00	2.59	28030	2.59
testCaseD30	1834	210	323.59	2.60	28122	2.60

the number of activity occurrences (cf. Definition 10) of each instance (note that each activity occurrence leads to 2 events, start and end).

4.5. Analysis

In this section, the case study findings are interpreted with the goal of answering the considered research questions. Regarding MRQ1(1), the proposed conformance checking method behaves correctly, as shown in column %*Correct* of Table 2 (as mentioned, such column includes the percentage of instances for which the proposed conformance checking method behaved correctly). This way, MRQ1(1) can be answered as true. Moreover, regarding MRQ1(2) (which checks the efficiency of the proposed conformance checking method when solving problems of different complexity), as can be observed in Table 2, the execution times for the conformance checking method which are obtained for both compliant and non-compliant instances are rather low (less than 329 ms for all cases) for problems of medium-size (number of activity occurrences varying between 43 and 210, cf. column *#actOcc* of Table 2). Furthermore, the execution time remains quite stable regardless of the relations which are given between the activities, i.e., similar execution times are obtained for testCaseA, testCaseB, testCaseC, testCaseD. However, as the number of activity occurrences increases (cf. column *#actOcc* of Table 2), the execution times slightly increase as well (cf. column *checkTime* of Table 2). Therefore, the efficiency of the proposed conformance checking method when solving problems of different complexities can be considered rather good, and hence, MRQ1(2) can be answered as true. Therefore, the first main research question (i.e., MRQ1, which checks the suitability of the proposed approach for carrying out the conformance checking of process instances according to a given data constraint-based process model) can be answered as true.

Regarding MRQ2(1) and MRQ2(3), related to the effectiveness and the efficiency of the proposed diagnosis process through the determination of MUSes, we can see that such process behaves correctly since the values for OF_R and OF_M match for all the cases, as shown in Table 3. Hence, both MRQ2(1) and MRQ2(3) can be answered as true. Furthermore, regarding MRQ2(2), as can be observed in Table 3 (column $diagTime_R$), the execution times for the diagnosis process which is based on reified constraints are rather low (less than 324 ms for all cases) for problems of medium-size (number of activity occurrences varying between 43 and 210, cf. column *#actOcc* of Table 3). Related to MRQ2(4), as can be observed in Table 3 (column $diagTime_M$), however, the execution times for the diagnosis process which is based on MUSes are much larger, although still they can be considered acceptable (less than 29s for all cases). Moreover, in general for both techniques, the execution time remains quite stable regardless of the relations which are given between the activities (i.e., similar execution times are obtained for testCaseA, testCaseB, testCaseC, testCaseD) and of the number of constraints which should be relaxed/removed (columns OF_R and OF_M). However, as the number of activity occurrences increases (cf. column *#actOcc* of Table 3), the execution times slightly increase as well (cf. columns $diagTime_R$ and $diagTime_M$ of Table 3). Therefore, the efficiency of the proposed diagnosis methods when solving problems of different complexity can be considered rather good, and hence, MRQ2(2) and MRQ2(4) can be answered as true. Regarding MRQ2(5), based on the results which are obtained for columns $diagTime_R$ and $diagTime_M$ of Table 3, we can affirm that the diagnosis method which is based on reified constraints is more efficient than the one based on MUSes in general regardless of the characteristics of the considered problems. This can be explained by the fact that the technique based on the attainment of the MUSes needs to compute the resolution of the CSP several times per trace in order to perform the search for a solution. Considering this, the second main research question (i.e., MRQ2, which checks the suitability of the proposed approach for carrying out the model-based fault diagnosis process by either using reified constraints or through the determination of MUSes) can be answered as true.

Additionally, it can be seen that, as expected, the diagnosis (cf. columns $diagTime_R$ and $diagTime_M$ of Table 3) takes longer than the conformance checking (cf. column *checkTime* of Table 2) since reaching an optimal solution in a COP (cf. Definition 7) is usually more time-consuming than checking the feasibility of a CSP (cf. Definition 6).

To summarise the conclusions of the empirical evaluation, the proposed approach (cf. Section 3) has been tested over a diversified set of data constraint-based process models (cf. Definition 2) and related instances (cf. Definition 5), since we considered:

- correct and representative data constraint-based process models,
- data constraint-based process models which cover all the complexity groups of filtering rules (models testCaseA, testCaseB, testCaseC, testCaseD),

- varied number of activity occurrences in the process instances (varying between 40 and 300 activity occurrences),
- varied activity durations (randomly generated between 1 and 10 to achieve diversified results), and
- both compliant and non-compliant instances which were randomly generated.

After applying the proposed approach (cf. Section 3) to such diversified set of data constraint-based process models and instances, results showed that:

- the proposed conformance checking method behaves correctly,
- the proposed diagnosis methods behave correctly,
- the execution times for the conformance checking method and for the diagnosis process which is based on reified constraints for both compliant and non-compliant instances are rather low,
- the execution times for the diagnosis process which is based on MUSes are much larger, although still they can be considered acceptable,
- the execution times for all the proposed methods remain quite stable regardless of the relations which are given between the activities, although as the number of activity occurrences increases, the execution times slightly increase as well,
- the diagnosis method which is based on reified constraints is more efficient than the one based on MUSes in general regardless of the characteristics of the considered problems, and
- as expected, the diagnosis takes longer than the conformance checking.

4.6. Plan validity

On the one hand, regarding the construct validity of the current case study, we consider that the proposed plan is suitable for reaching the considered purpose (i.e., analysing the effectiveness and the efficiency of the approach presented in Section 3 over a diversified set of data constraint-based process models, cf. Definition 2). To be more precise, the proposed response variables and the generated cases are considered adequate to answer the aforementioned research questions, as detailed in Sections 4.2 and 4.3. This way, the values which are obtained for such response variables when considering such cases are analyzed with the goal of interpreting the case study findings regarding the proposed research questions (cf. Section 4.5). Nevertheless, additional response variables could be defined and constraint-based process models generated to widen the analysis and the related findings.

On the other hand, regarding the external validity (i.e., the domain to which study findings can be generalized), since the considered cases are synthetic and not extracted from any real business with specific features, the study findings can be generalised to any domain which managed data constraint-based models of similar characteristics. Since a wide diversified set of different data-constraint based process models have been generated considering both correctness and representativeness (cf. Section 4.3), we consider that the results which are got are rather general. However, increasing the number and diversity of such models would further generalize the study findings.

5. Discussion and limitations

As mentioned, the proposed approach allows the BPs to be specified in a declarative way, which is an important step towards the flexible management of PAISs (van der Aalst et al., 2009). Moreover, since the considered declarative language is based on high-level constraints, the problems can be modelled in an easy way. This way, modelling business processes in Declare (Pesic, 2008) facilitates the human work which is involved in the process design & analysis phase at the same time as a higher flexibility is provided compared to imperative specifications.

While some works have been conducted on conformance checking for declarative business process models (De Leoni et al., 2012), in such works only control-flow perspective of BPs is typically considered although data perspective is also crucial when modelling and executing a process. Unlike existing proposals, our approach allows including business data rules in the process specification, which allows to decrease the cost related to the modification of its business logic, to shorten the development time, to externalize and easily share rules among multiple applications, and to make changes faster and with less risk (Weber et al., 2009).

In addition, unlike the current approach, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics (De Leoni et al., 2012; Burattin et al., 2012).

Furthermore, since the proposed approach is based on constraint programming, it allows for efficient conformance checking and diagnosis processes at the same time as facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities).

However, the proposed approach presents some drawbacks. First, the business analysts must deal with a non-standard language for the declarative specification of BPs, therefore a period of training is required to let the business analysts become familiar with Declare specifications. Moreover, the considered constraint-based models deal with both control-flow and data perspectives, but do not consider the resource perspective. It is intended to consider this aspect in our future work. As mentioned, in the empirical evaluation the effectiveness and efficiency of the proposed approach are analyzed over a diversified set of representative data constraint-based process models. However, additional evaluations of our proposal in the context of real process executions are desired and are intended to be addressed as future work.

6. Related work

As mentioned, there exist solid conformance checking techniques for imperative models (e.g., Rozinat and van der Aalst, 2008; Adriansyah et al., 2011; de Leoni and van der Aalst, 2013). As a major advantage of the proposed approach regarding these works, it considers declarative business process models, which facilitates the human work which is involved in the process design & analysis phase at the same time as a higher flexibility is provided compared to imperative models.

However, the application of conformance checking techniques for declarative models is not new. The most similar proposals in such respect are detailed as follows. Specifically, Grando et al. (2012, 2013) propose an approach for checking the conformance of computer interpretable guidelines based on a semantic-based approach that is fully independent of the language used for the specification of the guideline. Nevertheless, any diagnosis is provided. Similarly, the works (Pesic, 2008; De Leoni et al., 2012) present approaches for conformance checking over Declare models which are based on the generation of state automatons which represent exactly all traces that satisfy the LTL formulas of such models. In a similar way, (Burattin et al., 2012; de Leoni et al., 2014) propose an approach also based on the generation of such automatons but which additionally provides diagnosis. However, the big disadvantage of such approaches would be that the process of generating the automaton from the declarative specification is exponential with respect to the size of the LTL formulas (Montali et al., 2010). Moreover, unlike the current approach, in Burattin

et al. (2012) the diagnosis is carried out analysing each constraint in isolation instead of analysing the global set of constraints as a whole and providing the minimum set of constraints of the declarative models which need to be relaxed/removed to make such models feasible according to given instances.

In general, the approach presented in this paper differs from existing approaches on both imperative and declarative contexts in various ways. First, besides the control-flow constraints, our approach is able to deal with business data rules, which allows to decrease the cost related to the modification of its business logic, to shorten the development time, to externalize and easily share rules among multiple applications, and to make changes faster and with less risk (Weber et al., 2009). Secondly, the proposed approach is based on constraint programming, which allows for efficient conformance checking and diagnosis processes, and, due to its versatility, facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities). Moreover, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics (De Leoni et al., 2012; Burattin et al., 2012).

7. Conclusion and future work

To enhance the accurate management of flexible business processes, this work presents a constraint-based approach for performing conformance checking and for providing related diagnostics over declarative BP models which include both control-flow and data perspectives (by means of business data rules). With this aim, efficient constraint-based approaches are proposed to allow for an increased effectiveness when performing conformance checking as well as when dealing with the diagnosis. Specifically, for the conformance checking process, one constraint-based approach which includes the modelling of the declarative process models as constraint satisfaction problems (CSPs) as well as a set of global constraints implemented through filtering rules is proposed. For the diagnosis process, such CSPs are transformed to Max-CSPs (i.e., CSPs which allow the violation of some constraints) to identify the minimum set of constraints of the declarative models which need to be relaxed/ removed to make such model feasible according to given process instances. For solving such Max-CSPs two different techniques (i.e., (1) using reified constraints and (2) attaining the Minimal Unsatisfiable Subsets) are considered.

To demonstrate both the effectiveness and the efficiency of the proposed approach, the analysis of different performance measures related to a wide and diversified set of test models of varying complexity has been performed, concluding that the execution times which are obtained for both conformance checking and diagnosis are rather low for all the tested cases.

The proposed approach differs from existing approaches in various ways. Specifically, besides the control-flow constraints, our approach is able to deal with business data rules. Moreover, since the proposed approach is based on constraint programming, it allows for efficient conformance checking and diagnosis processes as well as, due to its versatility, facilitates the management of many aspects related to BPs which are not usually considered in previous work (e.g., dealing with non-atomic activities).

Summarising, the strengths of the proposed approach are: (1) it allows the BPs to be specified in a declarative way, which is an important step towards the flexible management of PAISs, (2) the problems can be modelled in an easy way, (3) unlike existing proposals, our approach allows including business data rules in the process specification, (4) unlike the current approach, most existing approaches exclusively focus on determining whether a given process instance conforms with a given process model or not without providing any detailed diagnostics, and (5) since the proposed approach is based on constraint programming, it allows for efficient conformance checking and diagnosis processes at the same time as facilitates the management of many aspects related to BPs which are not usually considered in previous work. However, the proposed approach also presents some weaknesses: (1) the business analysts must deal with a non-standard language for the declarative specification of BPs, therefore a period of training is required, (2) the considered constraint-based models deal with both control-flow and data perspectives, but do not consider the resource perspective, and (3) additional evaluations of our proposal in the context of real process executions are desired.

As for future work, it is intended to apply our proposal to actual processes from different domains which present flexible nature and where the accurate management of BPs is crucial (e.g., medical guidelines, processes from automotive industry and flight planning). Furthermore, some extensions of Declare templates are intended to be considered, e.g., branched templates (i.e., high-level relationships given between more than two activities). Additionally, we intend to consider the resource perspective in our future work.

8. Acronyms

AI	Artificial Intelligence
BNF	Backus Normal Form
BP	Business Process
BPM	Business Process Management
COP	Constraint Optimization Problem
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
LTL	Linear Temporal Logic
Max-CSP	Maximal Constraint Satisfaction Problem
MRQ	Main Research Question
MUSes	Minimal Unsatisfiable Subsets
OCT	Overall Completion Time
PAISs	Process-Aware Information Systems

Acknowledgment

This work has been partially funded by the Spanish Ministerio de Ciencia e Innovaci'on (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.eswa.2014.03.010.

References

Adriansyah, A., Dongen, B. F., & Aalst, W. M. P. (2011). Towards robust conformance checking. *Business process management workshops* (Vol. 66, pp. 122–133). Berlin Heidelberg: Springer.

Barba, I., & Del Valle, C., 2011. Filtering rules for condec templates – pseudocode and complexity. [Online; <http://www.lsi.us.es/quivir/irene/FilteringRulesforConDecTemplates.pdf>; accessed 24-January-2014].

Barba, I., & Del Valle, C., 2011. A constraint-based approach for planning and scheduling repeated activities. In *Proc. COPLAS* (pp. 55–62).

Barba, I., Del Valle, C., Weber, B., & Jiménez-Ramírez, A. (2013a). Automatic generation of optimized business process models from constraint-based specifications. *International Journal of Cooperative Information Systems, 22*(2), 1350009.

Barba, I., Weber, B., Del Valle, C., & Jiménez-Ramírez, A. (2013b). User recommendations for the optimized execution of business processes. *Data & Knowledge Engineering, 86*, 61–84.

Borrego, Diana, Eshuis, Rik, Gómez-López, María Teresa, & Gasca, Rafael M. (2013). Diagnosing correctness of semantic workflow models. *Data & Knowledge Engineering, 87*, 167–184.

Borrego, D., Gasca, R. M., Gómez-López, M. T., & Parody, L., 2010. Contract-based diagnosis for business process instances using business compliance rules. In *Proc. DX'10* (pp. 169–176).

Brereton, P., Kitchenham, B., Budgen, D., & Li, Z., 2008. Using a protocol template for case study planning. In *Proc. of EASE 2008*. BCS-eWiC.

Burattin, A., Maggi, F., van der Aalst, W. M. P., & Sperduti, A., 2012. Techniques for a posteriori analysis of declarative processes. In *Proc. EDOC 2012* (pp. 41–50).

de Kleer, J., & Kurien, J., 2003. Fundamentals of model-based diagnosis. In *Proc. Safeprocess 03* (pp. 25–36).

de Leoni, M., and van der Aalst, W. M. P., 2013. Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In *BPM* (pp. 113–129).

De Leoni, M., Maggi, F., & van der Aalst, W. M. P., 2012. Aligning event logs and declarative process models for conformance checking. In *LNCS: Vol. 7481* (pp. 82–97).

de Leoni, M., Maggi, F. M., & van der Aalst, W. M. P. (2014). An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. *Information Systems* (0).

Dumas, M., van der Aalst, W. M. P., & ter Hofstede, A. H. (Eds.). (2005). *Process-aware information systems: Bridging people and software through process technology*. Hoboken, NJ: Wiley-Interscience.

Dynadec. Comet downloads, 2010. [Online; <http://dynadec.com/support/downloads/>; accessed 20-January-2011].

Gasca, R. M., Valle, C., Gómez-López, M. T., & Ceballos, R. (2007). Nmus: Structural analysis for improving the derivation of all muses in overconstrained numeric csps. In *Current topics in artificial intelligence. Lecture Notes in Computer Science* (Vol. 4788, pp. 160–169). Berlin Heidelberg: Springer.

Grando, M. A., van der Aalst, W. M. P., and Mans, R. S., 2012. Reusing a declarative specification to check the conformance of different CIGs. In *LNBIP 100 (Part II)* (pp. 188–199).

Grando, M. A., Schonenberg, M. H., & Aalst, W. (2013). Semantic-based conformance checking of computer interpretable medical guidelines. In *Biomedical engineering systems and technologies. Communications in computer and information science* (Vol. 273, pp. 285–300). Berlin Heidelberg: Springer.

Huang, Z., van der Aalst, W. M. P., Lu, X., & Duan, H. (2010). An adaptive work distribution mechanism based on reinforcement learning. *Expert Systems with Applications, 37*(12), 7533–7541.

Kumar, V. (1992). Algorithms for constraint satisfaction problems: A survey. *AI Magazine, 13*(1), 32–44.

Lanz, A., Weber, B., & Reichert, M. (2012). Time patterns for process-aware information systems. *Requirements Engineering*.

La Rosa, M., Reijers, H. A., van der Aalst, W. M. P., Dijkman, R. M., Mendling, J., Dumas, M., et al. (2011). Apromore: An advanced process model repository. *Expert Systems with Applications, 38*(6), 7029–7040.

Liao, Shu-Hsien, Chu, Pei-Hui, & Hsiao, Pei-Yuan (2012). Data mining techniques and applications a decade review from 2000 to 2011. *Expert Systems with Applications, 39*(12), 11303–11311.

Montali, M. 2009. Specification and verification of declarative open interaction models: A logic-based approach (PhD thesis). Department of Electronics, Computer Science and Telecommunications Engineering, University of Bologna.

Montali, M., Pesic, M., van der Aalst, W. M. P., Chesani, F., Mello, P., & Storari, S. (2010). Declarative specification and verification of service choreographies. *ACM Transactions on the Web, 4*(1). art. no. 3.

Pesic, M., 2008. Constraint-based workflow management systems: Shifting control to users (PhD thesis), Eindhoven: Technische Universiteit Eindhoven.

Reichert, M., & Weber, B. (2012). *Enabling flexibility in process-aware information systems*. Springer.

Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of constraint programming*. Elsevier.

Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems, 33*(1), 64–95.

van der Aalst, W. M. P. (2011). *Process mining – discovery, conformance and enhancement of business processes*. Springer.

van der Aalst, Wil M. P., et al., 2011. Process mining manifesto. In *Business Process Management Workshops (1): Vol. 99* (pp. 169–194).

van der Aalst, W. M. P., Pesic, M., & Schonenberg, H. (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science – Research and Development, 23*(2), 99–113.

Van Hentenryck, P. (1989). *Constraint satisfaction in logic programming*. MIT Press.

Weber, B., Sadiq, S. W., & Reichert, M. (2009). Beyond rigidity – dynamic process lifecycle support. *Computer Science – Research and Development, 23*(2), 47–65.

Weske, M. (2007). *Business process management: Concepts, methods, technology*. Springer.