

Prognosing the Compliance of Declarative Business Processes Using Event Trace Robustness

María Teresa Gómez-López¹, Luisa Parody¹, Rafael M. Gasca¹,
and Stefanie Rinderle-Ma²

¹ Language and Computer System Department, University of Seville
Avda. de la Reina Mercedes s/n, 41012, Sevilla, Spain
`{maytegomez,lparody,gasca}@us.es`

² University of Vienna, Faculty of Computer Science, Austria
Währingerstrasse 29, 1090 Wien, Austria
`stefanie.rinderle-ma@univie.ac.at`

Abstract. Several proposals have studied the compliance of execution of business process traces in accordance with a set of compliance rules. Unfortunately, the detection of a compliance violation (diagnosis) means that the observed events have already violated the compliance rules that describe the model. In turn, the detection of a compliance violation before its actual occurrence would prevent misbehaviour of the business processes. This functionality is referred to as proactive management of compliance violations in literature. However, existing approaches focus on the detection of inconsistencies between the compliance rules or monitoring process instances that are in a violable state. The notion of robustness could help us to prognosticate the occurrence of these inconsistent states in a premature way, and to detect, depending on the current execution state of the process instance, how “close” the execution is to a possible violation. On top of being able to possibly avoid violations, a robust trace is not sensitive to small changes. In this paper we propose the way to determine whether a process instance is robust against a set of compliance rules during its execution at runtime. Thanks to the use of constraint programming and the capacities of super solutions, a robust trace can be guaranteed.

Keywords: Declarative Business Process, Compliance Rules, Model-based Prognosis, Robustness.

1 Introduction

Monitoring the compliance of process instances with a set of compliance rules has been studied by several approaches, e.g., [1,2,3,4,5]. According to [3], the differences between reactive management and proactive management of compliance violations can be distinguished. On the one hand, reactive management means to diagnose the execution of a process instance for compliance violations.

On the other hand, proactive management refers to the prognosis of upcoming violations. In general, the main problem with diagnosis is that when the malfunction is detected, it is too late to avoid the non-compliance. Hence, when monitoring process compliance, prognosis of compliance violations before their actual occurrence seems to be promising. Existing approaches focus on the detection of inconsistencies between the compliance rules [1], or on monitoring process instances that are in a violable state [5].

In [4], a method to diagnose the inconsistencies between compliance rules of the model and the events produced by the process execution is proposed. That paper also includes a method to inform about the correct interval of time where the activities should have been occurred. Unfortunately, when these compliance violations are found, it means that the misbehaviour has occurred or is going to happen inevitably. In this paper, we propose how to avoid the misbehaviour by prognosing that an incorrect behaviour is close to happen, and by considering that we are on time to avoid the non-conformity. We extend and adapt the model-based prognosis using robustness to declarative business process monitoring. Model-based prognosis compares the expected compliance rules with the received event traces, including the analysis that small problems can occur in the future.

The advance knowledge of possible compliance violations empowers the execution to be more robust to malfunctions. *Robustness* can be understood as 'the ability of a system to cope with misbehaviours during execution'. On top of being able to avoid possible violations, a robust trace is not sensitive to small changes. The meaning of small changes depends on the type of problem, in our case it means possible delays in the execution of the activities. One way to solve the management of robustness found in the literature is by means of the use of (a-b)-super solutions [6]. We propose to adapt the key idea of the super solutions by means of applied it to compliance analysis in declarative business processes. We use the super solutions to know the possibility to find different alternatives of execution for the events in the future. This analysis helps us to be sure that, if a small unexpected behaviour occurs, we can find another consistent trace to the compliance rules. If it is not possible to find super solutions, it means that the trace is not robust to small changes, and an incorrect behaviour can occur in a more easy way. In order to achieve the prognosis of the compliance of declarative business processes in an automatic way, we propose in this paper:

- **Describe the Model-based Prognosis and the event trace model.** It implies to describe the declarative business process by means of compliance rules, and the observational model by means of an event trace. In order to describe the declarative business process, we use an extension of Declare [7] inspired in the Mobucon monitoring framework [8][9]. The proposed extension is derived from the example used in this paper, although the extension itself is not the aim of the paper.
- **Prognosticate the non-compliance using Event Trace Robutness and Find Critical Activities.** In order to determine a potential non-compliance of the business rules, the prognosis is performed. Using the robustness analysis of the event trace for a model, it is possible to detect a

non-conformity before it occurs. In order to perform it, we propose a computational model that verify, prognosticate, and determine the critical activities to maintain the conformity automatically. The proposal uses Constraint Programming paradigm. If a non-robust trace is found, since one of the activities can only be executed in an instant, then these activities need to be determined to advise the user about the importance to execute these activities in an instant, avoiding a non-conformity in the future.

The paper is structured as follows: Section 2 presents an example where a declarative model and a trace of events are shown, we have included what we expect from the prognosis analysis. Section 3 includes the necessary definitions to formalize and clarify the proposal. Section 4 analyses the necessary computational models to automatize the prognosis based on event trace robustness. Section 5 details the verification method, the prognosis process and the determination of the set of critical activities, using the approach explained in Section 4. In Section 6, the evaluation times and the complexity of resolutions are studied for an extended and real example. Section 7 analyses the previous related work found in the literature. Finally, conclusions and future works are presented.

2 Motivating Example

In order to motivate the importance to detect an inconsistency before it occurs, we use as example the protocol of pregnancy of the Health System of Andalusia. The example highlights the necessity to execute a set of activities in specific time intervals, and in accordance with other activities, due to some tests are related to the size of the fetus and/or the legal connotations of the gestation week. Although the whole example is evaluated in Section 6, only a fragment of the model is shown in Figure 1. To model the example, we have decided to used Mobucon with the time patterns extension of Declare, since it can fit the necessities of the example better than others. But we need to extent Mobucon framework, to model the interval of time where the activities can be executed and the time needed between the activities. The problem of modelling health protocols is an open problem in the area, as explained in [10]. The objective of our proposal is not based on the definition of a new language neither to extend the existing, only the establishment of the necessary patterns for the problem requirements.

Figure 1 represents nine activities where: one of them represents the initialization of the process (Amenorrhoea); the activities execution order is described by means of precedence, response and succession relations; possible time interval constraints associated to the activities, and; maximum and minimum time distance between the execution of two activities. Also derived from the example, we assume that the execution of an activity is only represented by means of an occurrence in an instant (one event), not being necessary to represent or treat, the duration of the activities.

In order to understand the idea of prognosis, and its relation with the robustness, let us include the three possible reports and some examples of the

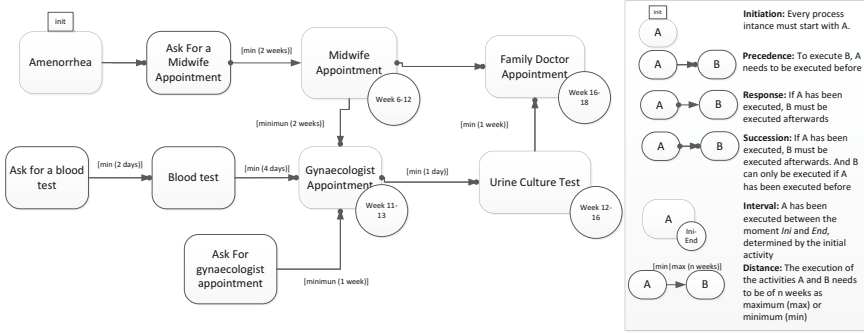


Fig. 1. Pregnancy example described by the Declare extension

corresponding traces, that our proposal can return during the monitoring, being each event described by means of the tuple $\langle \text{Time}, \text{Activity} \rangle$.

- **A correct and robust trace of events.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 \text{ (week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 \text{ (week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 80 \text{ (week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 \text{ (week } 12) \rangle\}$, is correct and robust since every non executed activities can be executed in more than one instant in the future.
- **A correct although weak trace, since activities A_m, \dots, A_n can be only executed in one time instant in the future.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 \text{ (week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 \text{ (week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 81 \text{ (week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 \text{ (week } 12), \text{Ask for blood test} \rangle, \langle \text{Day } 87 \text{ (week } 13), \text{Blood test} \rangle\}$, is correct although weak, since activity *Gynaecologist Appointment* can only occur in the day 91, then if there is some problem that day, a non-conformity will occur.
- **An incorrect event trace where the events e_m, \dots, e_n are the responsible of the inconsistency.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 \text{ (week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 \text{ (week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 81 \text{ (week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 \text{ (week } 12), \text{Ask for blood test} \rangle, \langle \text{Day } 87 \text{ (week } 13), \text{Blood test} \rangle, \langle \text{Day } 92 \text{ (week } 14), \text{Gynaecologist Appointment} \rangle\}$, is incorrect since the event $\langle \text{Day } 92 \text{ (week } 14), \text{Gynaecologist Appointment} \rangle$ was executed too late.

3 Model-Based Prognosis in Declarative Business Processes

Compliance monitoring tends to be based on the comparison of the events that describe the execution of the process instances, and the compliance rules that define the policies to comply [3]. Unfortunately, the determination of a compliance

violation (diagnosis) means that the process execution has already produced an incorrect behaviour. In order to avoid a non-compliance, the prognosis detects signs of a possible misbehaviour before it occurs. We propose a computational model that warns about the critical activities during the execution of a process, to avoid the violation of the compliance in the future.

The architectures that support the monitoring has generally the following five modules: Process Execution (CRM, ERP, ...); Event Services; Compliance Monitoring; Compliance Requirements, and; Reporting and visualization of the compliance process. Since our proposal focuses on the prognosis of the non-compliance, the contribution is located in the Compliance Monitoring layer, where various activities are developed to support the prognosis of a declarative process model in function of the compliance requirements specified, and the events observed. The necessary activities to prognosticate (shown in Figure 2) are: Verification, Diagnosis, Prognosis based on Robustness, and Critical Activities Analysis. Each activity is related to the type of report that the proposal can offer: verify the correctness, diagnose an inconsistency, prognosticate a misbehaviour, or find the critical activities. Since how to verify and diagnose the non-compliance events was studied in a previous work [4], the two other tasks are faced in this paper, although to help in the prognosis, some improvements are included in the verification task.

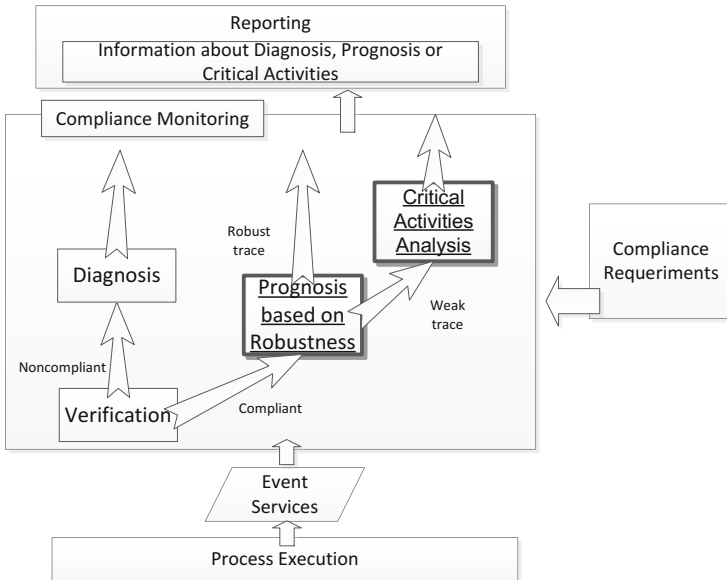


Fig. 2. Business Rule Monitoring Architecture

In order to describe the model that will be prognosticated, our approach adapts the definition of model-based diagnosis [11] to prognose declarative business processes.

Definition 1 (Prognosis Model (ProMod)). *It is formed by:*

- **Declarative BP Model (System Description):** *A declarative model is formed of a set of activities and a set of compliance rules. The compliance rules describe the relational order between the activities (precedence, response, succession), time intervals of execution associated to the activities ([Ini-End]), and time distance between the activities ([min | max (time)]).*
- **Event Trace (Observational Model).** *The events $\{e_1, \dots, e_p\}$ that make visible out of the process the execution of the activities between the initial time and the current time. Each event is associated to one and only one activity, although an activity can be executed several times and represented by several events.*

During the monitoring of a process execution, several event traces are observed. We assume that the events of these traces can be differentiated, being possible to analyse each trace separately. Each trace of events (partial or full) can be compliant or not, according to the compliance rules of the declarative BP model (Definition 1). A compliant partial instance can be *robust* or *weak*. In order to clarify all these terms, we include the following definitions:

Definition 2 (Full Trace (FT) and Partial Trace (PT)). *Being a trace A formed of the set of events $\{e_1, \dots, e_m\}$, where the set $Executed(A)$ represents the activities, without repetition, executed in the trace A . A is a Full Trace if there not exists a compliant trace B , such that $Executed(A) \subset Executed(B)$. In the opposite way, A is a Partial Trace iff there exists a compliant trace B such that $Executed(A) \subset Executed(B)$.*

Definition 3 (Promising Trace (PrT), Robust Trace (RT) and Weak Trace (WT)). *Being A a partial trace formed of the list of events $\{e_1, \dots, e_m\}$, A is a:*

- **Promising Trace**, *if A is a compliant PT for a Declarative BP Model.*
- **Robust Trace**, *if A is a PrT and, for all the non-executed activities of A according to its FT ($Executed(B) - Executed(A)$), there exists more than one instant to execute these non-executed activities. Formally, being: 'a' an activity; A a PrT, and; B and C , two FTs of A . A is a RT, iff $\forall a \in \{Executed(B) - Executed(A)\}$, there exists another FT C , where the only one difference between B and C is the execution time of the activity 'a' (t_B and t_C) respectively. The distance between the two execution times ($|t_B - t_C|$) depends on the smallest time reference included in the ProMod.*
- **Weak Trace**, *if A is a PrT although not a RT.*

4 Automating Robust-Based Prognosis Using Constraint Programming

In order to perform automatically the prognosis, and determine the critical activities of a declarative model, we propose the use of the concept of *super solution* of Constraint Programming. The use of Constraint Programming paradigm brings major advantages, such as early identification of non-compliance for a PT (even before all the activities are executed) [12]. In addition, several algorithms and tools have been developed, which can efficiently solve the model. In order to transform the ProMod into a constraint problem, it is necessary to explain: what is Constraint Programming, and how we can use it in model-based prognosis (Subsection 4.1); how to model the trace of events by means of constraints (Subsection 4.2), and; how to transform the declarative BP model into a CSP (Subsection 4.3).

4.1 Constraint Programming and Prognosis Using Robustness

With the aim of prognosticate possible inconsistencies using event trace robustness, we propose the use of the concept of super solutions of Constraint Programming [6]. An (a-b)-super solutions is a solution in which if the values assigned to a variables are no longer available, the solution can be repaired by assigning these a variables with new values and at most b modifications in other variables. To find a super solution guarantees the existence of a small set of repairs when the future changes in a small way. To be applied in prognosis, we use (1,0)-super solutions, that follows Definition 3 of RT and WT , where if one variable loses its value, we can find another solution by re-assigning this variable with a new value, without the modification of another variable.

To find the existence of (1-0)-super solutions in accordance to a PT , Constraint Satisfaction Problems (CSPs) can be applied.

CSPs represent a reasoning methodology consisting of a model a problem formed by variables, domains and constraints. Formally, it is defined as a triple $\langle X, D, C \rangle$ where $X = \{x_1, x_2, \dots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \dots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \dots, C_m\}$ is a set of constraints. A constraint $C_i = (V_i, R_i)$ specifies the possible values of the variables in V simultaneously to satisfy R . Usually, to solve a CSP, a combination of search and consistency techniques is used [13].

When an objective function f has to be optimized (maximized or minimized), then a Constraint Optimization Problem (COP) is used, which is a CSP with an objective function f .

The specific CSP or COP created to automate the verification and the prognosis, will be detailed in Section 5. Before, it is necessary to know how to model the ProMod by means of constraints, to know after, how these constrains can be combined in a CSP.

4.2 Modelling Event Trace by Means of Constraints

As it was introduced in Definitions 2 and 3, to know if a *PT* is compliance or not, the activities that can be executed in the future (*FT*) need to be analysed. For this reason, it is necessary to distinguish between events that have already happened (observational model) and the events that will be thrown by the process execution (events of the future) [4]. For example, analysing the compliance rule that relate activities *Midwife Appointment* and *Family doctor Appointment*, if an event related to the activity *Midwife Appointment* is executed, *Family doctor Appointment* must be executed afterwards. The model needs to support the representation of the executed events, for example $\langle 52, \text{Midwife Appointment} \rangle$, and the events expected in the future, such as $\langle t_x, \text{Family doctor Appointment} \rangle$, where t_x represents an instant in the future ($t_x > \text{current Time}$) where the activity must occur to satisfy the model.

In order to capture the information on events of the past and the possibilities in the future, the model-based prognosis for each event (executed or not) must include a variable associated to the timestamp [4]: the pair of variables $\langle \text{Executed}, \text{Time} \rangle$ represents if the event has been executed with the Boolean variable *Executed*, and the instant when it was executed with the numerical variable *Time*. In particular, the following rules depending on the execution time of each event must be satisfied [4]:

- If the event e has been executed: $\text{Executed} = \text{true} \wedge \text{Initial Time} \leq \text{Time} \leq \text{Current Time}$.
- If the event e has not been executed but will be executed in the future: $\text{Executed} = \text{false} \wedge \text{Time} > \text{current Time}$.
- If the event e has not been executed and will not be executed in the future: $\text{Executed} = \text{false} \wedge \text{Time} = -1$.

4.3 Transforming Declarative BP Model into Numerical Constraints

In order to model the compliance rules of the *ProMod*, it is necessary to transform the patterns included in Figure 1 into computable constraints. Based on the model that describes the execution of the events that represents the activities, how the activity relations can be represented by means of numerical constraints is depicted in Figure 3. These transformations define, by means of numerical constraints, what each compliance rule means.

5 Verification of Compliance, Prognosis and Critical Activities Determination

The various activities related to model-based prognosis are based on Constraint Programming to determine if the event traces analysed during the monitoring are: *PrTs*, *RTs* or *WTs*. The example shown in Figure 4 represents a *PT*, where the various events located above the activities represent the execution of the activities of the declarative model. Thanks to the use of Constraint Programming,


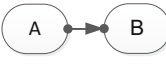
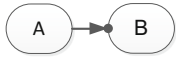

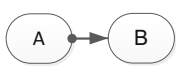

Template	Logical Formula	Template	Logical Formula
	$A_{EX} \wedge A_T = 0$		$A_{EX} \rightarrow B_{EX} \wedge A_T < B_T$ $\wedge B_{EX} \rightarrow A_{EX} \wedge A_T < B_T$
	$B_{EX} \rightarrow A_{EX} \wedge A_T < B_T$		$A_{EX} \rightarrow A_T \geq \text{Ini}$ $A_T \leq \text{End}$
	$A_{EX} \rightarrow B_{EX} \wedge A_T < B_T$		min: $A_{EX} \wedge B_{EX} \rightarrow$ $A_T + \eta \leq B_T$ max: $A_{EX} \wedge B_{EX} \rightarrow$ $A_T + \eta \geq B_T$

Fig. 3. Patterns of transformation from declarative BP model to numerical constraints

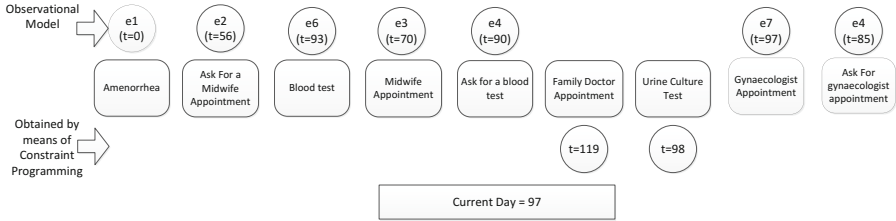


Fig. 4. Example of Trace

it is possible to know if this PT is a PrT . According to CSP, a PT is a PrT if it is possible to infer events for the non-executed activities (events below the activities), to obtain a FT . Following, how to ascertain if the event trace is a PrT , a WT or a RT is detailed.

5.1 Compliance Verification Using Constraint Programming

In order to compute the verification of a trace (PrT), we propose the creation of a correspondence between the elements of the $ProMod$ and the parts of the CSP as follows:

- *Variables* (X) is formed by the variables that represent the execution and the instant of execution of the activities (A_{Ex} and A_T) for each activity of the model.

- *Domains (D)* is defined as Boolean for the variables A_{Ex} , and Integer for the variables A_T . A_T is represented by means of the smallest time reference included in the *ProMod*, days in our example. It is related to the quality of the event stream to be analyzed, in particular, the granularity of the recorded events [14]. Recording of events at a too coarse granularity might lead to imprecise or even incorrect results, e.g., in the medical domain [15]. The same holds for compliance monitoring and prognosis. In order to address this challenge, the proposed approach might be combined with techniques for enhancing event quality as proposed, for example, in [15].
- *Constraints (C)* is the set of compliance rules represented by means of numerical constraints, defined over the variables A_{Ex} and A_T , and derived from the transformation explained in Figure 3. Moreover, it is necessary to assign: the rules that relate the values between each pair of variables (A_{Ex} , A_T), as explained in Subsection 4.2; the values of the observed events to the variables A_{Ex} and A_T , and; the current time of the system to differentiate past and future events.

If a tuple of values for the variables X is found in the domain D , where all the constraints C are satisfiable, then we can assure that the trace is a *PrT*. In the opposite way, a diagnosis process will be necessary to find the event or events responsible of the misbehaviour. Table 1 shows a fragment of the CSP for the trace of Figure 4. To solve the CSP implies to solve the robustness problem.

A singularity of the declarative models is the possible representation of deviations from the prescribed model. This deviation might happen due to several reasons, such as exceptional situations, for the example, it might be necessary to see the midwife twice during week 6 to 12 (see Figure 1). This results in the multiple occurrence of activities [16] or Multiple Instance Patterns¹. The prognosis approach could deal with multiple activity occurrence by updating the related CSP once the occurrence of a multiple activity instantiation is detected.

5.2 Prognosis Using Event Trace Robustness and the Critical Activities Determination

Definition 3 of *RT* is aligned with the (1-0)-super solution in the sense that, a *PrT* is a *RT*, iff there exists a *FT*, such that every non-executed activities can be executed in at least two different instants of time. Therefore, to ascertain if an event trace is a *RT* or *WT*, it is necessary to find the *FT*, and the CSP of Table 1 needs to be modified to take into account various aspects:

- **How is the *FT* obtained?:** Since there are several *FT* that can satisfy a *PrT*, to perform the best prognosis, we have to take into account the widest *FT* for a *PrT*, given that more activities will be analysed. It can be obtained including an objective function that maximize the number of activities executed (the summatory of the A_{Ex} variables of the activities of the model) (as shown in Table 2). The objective function transforms the CSP into a COP.

¹ <http://www.workflowpatterns.com/patterns/control/>

Table 1. Example of Constraint Satisfaction Problem for Verification

```

//Variables (Time and execution for each activity) and Domains:
AmenorrheaEx, AskForMidwifeAppointmentEx, BloodTestEx, ...: Boolean
AmenorrheaT, AskForMidwifeAppointmentT, BloodTestT, ...: Integer
...
//Initialization of variables
currentTime = 97
initialTime = 0
//Compliance rules of the model
Amenorrheat = initialTime
AskForMidwifeAppointmentEx → AmenorrheaEx
    ∧ Amenorrheat < AskForMidwifeAppointmentt
AskForMidwifeAppointmentEx → MidwifeAppointmentEx
    ∧ AskForMidwifeAppointmentt < MidwifeAppointmentt
MidwifeAppointmentEx → FamilyDoctorAppointmentEx
    ∧ MidwifeAppointmentt < FamilyDoctorAppointmentt
    ∧
FamilyDoctorAppointmentEx → FamilyDoctorAppointmentEx
    ∧ MidwifeAppointmentt < FamilyDoctorAppointmentt
...
//Events occurred
AmenorrheaT = 0
AskForMidwifeAppointmentT = 56
MidwifeAppointmentT = 70
...
//Time Interval Execution
MidwifeAppointmentEx → 36 ≤ MidwifeAppointmentT ≤ 84
FamilyDoctorAppointmentEx → 106 ≤ FamilyDoctorAppointmentT ≤ 126
...
//Distance between activities
AskForMidwifeAppointmentEx ∧ MidwifeAppointmentEx →
    AskForMidwifeAppointmentt + 14 ≤ MidwifeAppointmentt
BloodTestEx ∧ GynaecologistAppointmentEx →
    BloodTestt + 4 ≤ GynaecologistAppointmentt
...
//Constraints to describe time and event execution relation
(AmenorrheaEx ∧ InitialTime ≤ AmenorrheaT ≤ currentTime) ∨ ...
(AskForMidwifeAppointmentEx ∧ InitialTime ≤ MarkedAsClearedT ≤ currentTime) ∨ ...
(MidwifeAppointmentEx ∧ InitialTime ≤ MidwifeAppointmentT ≤ currentTime) ∨ ...
...

```

Table 2. Finding the widest FT for the PT

```
//Objective Function
Maximize (AmenorrhoeaEx + ... + AskForMidwifeAppointmentEx)
```

- **Where (in which domain) is it possible to find another solution for a (1-0)-super solution?:** It will depend on the *rhythm* of the process, and the frequency of process monitoring: one per day in our example. But in general, it should be done after the current time, since it is not possible that an activity that has not occurred yet, takes an execution time of the past.
- **Which variables can 'lose' their values?:** In our case, the variables that can lose their values, according to the (1-0)-super solutions, will be the non-executed activities (*Executed(FT)-Executed(PrT)*), since the rest have already been executed in the past. The problem can appear when one activity executed in the future has only one day that satisfy the compliance rules, and finally this day occurs a problem, not being a robust solution. In order to know if a trace is a *RT*, we propose to include a variable in the CSP to represent that an activity can be executed in two different instants. The idea of the modification is shown in Table 3. If a solution is found, it is a (1-0)-super solution related to the variable a_t , since although this variable loses its value, it is possible to find another value a'_t that satisfy the compliance rules.

Table 3. Finding the widest FT for the PT

```
//Variables and Domain
at, bt, ct, a't
//Constraints
a+b=c
a'+b=c
at<a't ∧ currentTime < a't
```

An example of what it is necessary to include in the CSP of the example in Table 1, is the shown in Table 4, to analyse the robustness of *Urine Culture Test* in this case. If a tuple of values for *Urine Culture Test* is found, then it means that there is a (1-0)-super solution for the activity *Urine Culture Test*. In this CSP, it is also included the maximum number of activities to executed for a *FT*, value obtained in the COP of the verification process, nine in this example.

Since it is possible to find *WTs*, it is useful for the process execution to know the possible conflicts activities to be sure that they are executed in the unique instant that keep the compliance of the compliance rules. It means to inform about the variables that represents the events that imply the execution of the activities, that has no (1-0)-super solutions. The information of the Critical Activities is directly obtained from the evaluation of each CSP created as explained in Table 4.

Table 4. Example of Constraint Satisfaction Problem for Robustness Analysis

```

//Variables and Domains:
..., UrineCulturet, UrineCulture't: Integer
//Compliance rules of the model
...
FamilyDoctorAppointmentEx → UrineCultureEx
    ∧ UrineCulturet < FamilyDoctorAppointmentt
FamilyDoctorAppointmentEx → UrineCultureEx
    ∧ UrineCulture't < FamilyDoctorAppointmentt
GynaecologistAppointmentEx → UrineCultureEx
    ∧ GynaecologistAppointmentt < UrineCulturet
∧
UrineCultureEx → GynaecologistAppointmentEx
    ∧ GynaecologistAppointmentt < UrineCulturet
GynaecologistAppointmentEx → UrineCultureEx
    ∧ GynaecologistAppointmentt < UrineCulture't
∧
UrineCultureEx' → GynaecologistAppointmentEx
    ∧ GynaecologistAppointmentt < UrineCulture't
//Time Interval Execution
...
UrineCultureEx → 78 ≤ UrineCultureT ≤ 112
UrineCultureEx → 78 ≤ UrineCulture'T ≤ 112
//Distance between activities
...
GynaecologistAppointmentEx ∧ UrineCultureEx →
    GynaecologistAppointmentt + 1 ≤ UrineCultureT
GynaecologistAppointmentEx ∧ UrineCultureEx →
    GynaecologistAppointmentt + 1 ≤ UrineCulture'T
UrineCultureEx ∧ FamilyDoctorAppointmentEx →
    UrineCulturet + 7 ≤ FamilyDoctorAppointmentT
UrineCultureEx ∧ FamilyDoctorAppointmentEx →
    UrineCulture't + 7 ≤ FamilyDoctorAppointmentT
//Events occurred
...
//Constraints to describe time relation and event Execution
...
//To find (1-0)-super solutions
(UrineCultureT < UrineCulture'T) ∧ (currentTime < UrineCulture'T)
//For the FT
(AmenorrhoeaEx + AskForMidwifeAppointmentEx + BloodTestEx + ... = 9)

```

6 Evaluation of the Case Study

To analyse the applicability of our proposal, we have implemented the full example of pregnancy protocol. The whole model and some example of CSPs are available in <http://www.lsi.us.es/~quivir/index.php/Main/Prognosis>. It is formed of 28 activities, 33 relation orders (24 precedences, 7 responses, 2 successions), 12 interval relations, and 17 distance relations. In Figure 5 the execution times to (a) Verify the conformity of partial trace, and (b) Analyse the robustness of a partial trace, are shown. Each measurement of both graphics represents the Verification and Prognosis respectively, with the activities executed until the moment, for example: the first measurement represents that there is a *PT* formed of an event of *Amenorrhoea* activity, the second measurement represents that there is a *PT* formed of two events of *Amenorrhoea* and *Ask For Midwife Appoinmet*, . . . Both graphics show as soon as more variables are instantiated, the time decreases. This occurs since the number of variables to assign values and to verify is reduced in each test.

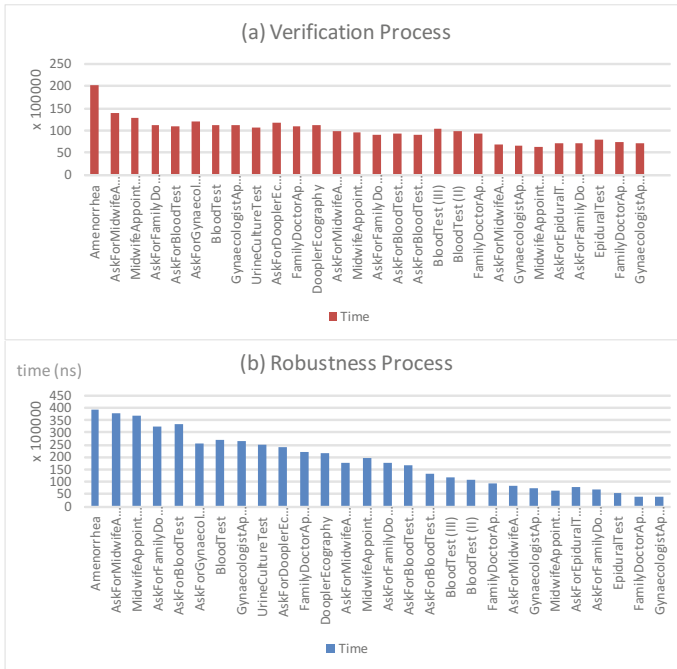


Fig. 5. Evaluation times for Verification and Robustness using Constraint Programming in the full example

7 Related Works

In this paper three challengers have been faced: how to model an example of a real medical guidelines; the use of the monitoring architecture, prognosis in our case, and; what technique can solve automatically the model-based prognosis. According to these challengers, the three corresponding areas have been studied.

Related to the languages of declarative models, several languages have been proposed to describe declaratively the ordering and temporal relations between activities in the business processes. Independently of the language, the common idea of declarative business process modelling is that a process is seen as a trajectory in a state space and that declarative constraints are used to define the valid movements in that state space [17]. The differences between declarative process languages are centred in the different perception of what is a state. Most relevant have been analysed in [18]. Some examples are the case handling paradigm [19], the constraint specification framework [20], the Declare language [21], PENELOPE language [22], Case Management (CMMN) [23], or EM-BrA²CE [24].

Related to the use of the monitoring architecture, it is possible to find works in literature related to verification, diagnosis, recommendations, . . . There are many proposals that use the compliance rules and the monitoring of events to verify the correctness of a business process instance [25,1,5]. As discussed in [3], the *proactive management* of compliance violations during process runtime constitutes an important functionality. Even though the evaluation of existing approaches provided in [3] shows that some of them support proactive management, these approaches focus on detecting violations caused by the interplay between compliance constraints. In this approach, by contrast, the focus is on foresee violations that might become very likely in the course of process execution. This specifically necessitates to incorporate a notion of metric time as well as time distances and deadlines for process activities. Referring to [3] again, it can be argued that none of the existing approaches supports these temporal specifications within compliance constraints. Hence, the contribution of the proposed solution is two-fold: a) incorporation of time information that is presented in many real-world application (see the pregnancy example in this paper, but also the skin cancer treatment example provided in [26]); b) introducing and monitoring the likeliness of upcoming compliance violations (robustness or weakness of compliance for process instances).

Conforti et al. in [27] establish a recommended system in order to predict possible risks in the model. The event log is analysed in order to study the sequence of activities executed and the specific values consumed and provided by these activities. This analysis establishes the occurrence of possible faults (over-time, reputation-loss, and cost overrun), and provide a decision support for risk reduction. However, the proposal needs the historical information extracted from the event logs in order to provide a solution. At the same time, Maggi et al. in [28] also establish a predictive system in order to prevent customer to execute an instance which is not going to obtain the desired business goal. The authors establish a system to recommend which activity must be performed and

what data input values must the customer provide, but it is not treat the time restrictions used in medical guidelines.

About the techniques, the analysed languages use different knowledge representation paradigms, that enable different types of compliance rule management. For instance, Declare language is expressed in Linear Temporal Logic (LTL), whereas the PENELOPE language is expressed in terms of the Event Calculus.

Linear Temporal Logic (LTL) expressions can be used to represent desirable or undesirable patterns. LTL formula can be evaluated by obtaining an automaton that is equivalent to the formula and checking whether a path corresponds to the automaton. Unfortunately, the use of automata does not allow to prognose according to time restrictions. It is due to our proposal does not only analyse the compliance rules activated because the antecedent was occurred. We include the whole model in the prognosis process as in [29], but with the difference that in this case a declarative language is used instead of an imperative language. On the other hand, the Event Calculus [30] is a first-order logic programming formalism that represents the time-varying nature of facts, the events that have taken place at given time points and the effect that these events reflect on the state of the system. Although one of the advantages of the use of event calculus is the ability to deductively reason about the effects of the occurrence of events and, more important is the abductive reasoning to discover a hypothesis about the malfunction to explain the evidence of events. But it does not have the capacity to propose a new set of data (events in this case) to avoid this misbehaviour, inferring possible misbehaviours in the future.

We have decided to use Constraint Programming since: it is a very mature area that has been applied to a wide range of problems, and with high level of complexity; it uses propagation techniques to reduce the search space in an efficient way; there are numerous tools and algorithms to model and solve problems, and; it permits an easy definition of the complex data using a wide range of constraints, such as implication constraints, disjunctive constraints, reified constraints, global constraints, and channelling constraints. Previous solution, as [29] [12], have used Constraint Programming to decision-making and diagnosis, respectively, over the input data of the process to avoid a failure, but not according to the execution moment of the activities, only for the input data of the activities, and in an imperative context.

8 Conclusions and Future Work

In this paper, we propose a framework to support model-based prognosis analysing the event trace robustness. According to the definition of robustness, we use the search of (1-0)-super solutions by using Constraint Programming. The (1-0)-super solutions can detect automatically the existence of critical activities, avoiding possible misbehaviours. It has been motivated by the use of a real example that represents the medical guidelines of the pregnancy protocol. The obtained evaluation times are very promising, thanks to use constraint programming.

As future work, we consider very interesting the analysis of: how the robustness can be related to the number of instances executed in each moment, since the systems have a limited number of resources, the distance between activities can be defined by the number of instances; how to facilitate the modification of the definition of robustness, to be implemented automatically, or; how it would be possible to define different degrees of robustness (low, medium, high, ...)

References

1. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
2. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the event calculus. ACM TIST 5(1), 17 (2013)
3. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., Aalst, W.M.P.v.d.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: 17th Int'l EDOC Conference (accepted for publication, 2013)
4. Gómez-López, M.T., Gasca, R.M., Rinderle-Ma, S.: Explaining the incorrect temporal events during business process monitoring by means of compliance rules and model-based diagnosis. In: EDOC Workshops, pp. 163–172 (2013)
5. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 82–99. Springer, Heidelberg (2011)
6. Hebrard, E., Hnich, B., Walsh, T.: Super solutions in constraint programming. In: Régin, J.-C., Rueher, M. (eds.) CPAIOR 2004. LNCS, vol. 3011, pp. 157–172. Springer, Heidelberg (2004)
7. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime Verification of LTL-Based Declarative Process Models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
8. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime verification of LTL-based declarative process models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
9. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: An approach based on colored automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
10. Dunkl, R., Fröschl, K.A., Grossmann, W., Rinderle-Ma, S.: Assessing medical treatment compliance based on formal process modeling. In: Holzinger, A., Simonic, K.-M. (eds.) USAB 2011. LNCS, vol. 7058, pp. 533–546. Springer, Heidelberg (2011)
11. Struss, P.: New techniques in model-based diagnosis. In: Ramani, S., Anjaneyulu, K.S.R., Chandrasekar, R. (eds.) KBCS 1989. LNCS, vol. 444, pp. 428–437. Springer, Heidelberg (1990)
12. Gómez-López, M.T., Gasca, R.M., Pérez-Alvarez, J.M.: Compliance validation and diagnosis of business data constraints in business processes at runtime. Information Systems. Elsevier (2014)
13. Dechter, R.: Constraint Processing. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann (May 2003)

14. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) *BPM Workshops 2011, Part I. Lecture Notes in Business Information Processing*, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
15. Binder, M., Dorda, W., Duftschmid, G., Dunkl, R., Fröschl, K.A., Gall, W., Grossmann, W., Harmankaya, K., Hronsky, M., Rinderle-Ma, S., Rinner, C., Weber, S.: On analyzing process compliance in skin cancer treatment: An experience report from the evidence-based medical compliance cluster (EBMC2). In: *Int'l Conference on Advanced Information Systems Engineering*, pp. 398–413 (2012)
16. Rinderle-Ma, S., Reichert, M., Jurisch, M.: On utilizing web service equivalence for supporting the composition life cycle. *International Journal of Web Services Research (IJWSR)* 8(1), 41–67 (2011)
17. Bider, I., Khomyakov, M., Pushchinsky, E.: Logic of change: Semantics of object systems with active relations. *Autom. Softw. Eng.* 7(1), 9–37 (2000)
18. Parody, L., Gómez-López, M.T., Gasca, R.M.: Data-oriented declarative language for optimizing business processes. In: *International Conference on Information System Development, ISD 2013*. Springer (2013)
19. van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. *Data and Knowledge Engineering* 53 (2005)
20. Sadiq, S.W., Orłowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. *Inf. Syst.* 30(5), 349–378 (2005)
21. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
22. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
23. OMG: Case management model and notation (cmmn). reference manual (2014)
24. Goedertier, S., Haesen, R., Vanthienen, J.: Em-bra²ce v0.1: A vocabulary and execution model for declarative business process modeling. *FETEW Research Report KBI_0728*, K.U.Leuven (2007)
25. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: de Lara, J., Zisman, A. (eds.) *FASE 2012*. LNCS, vol. 7212, pp. 146–162. Springer, Heidelberg (2012)
26. Dunkl, R., Fröschl, K.A., Grossmann, W., Rinderle-Ma, S.: Assessing medical treatment compliance based on formal process modeling. In: Holzinger, A., Simonic, K.-M. (eds.) *USAB 2011*. LNCS, vol. 7058, pp. 533–546. Springer, Heidelberg (2011)
27. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) *CAiSE 2013*. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013)
28. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. *CoRR* abs/1312.4874 (2013)
29. Gómez-López, M.T., Gasca, R.M., Pérez-Alvarez, J.M.: Decision-making support for the correctness of input data at runtime in business processes. *International Journal of Cooperative Information Systems* 23 (2014)
30. Kowalski, R.A., Sergot, M.J.: A logic-based calculus of events. *New Generation Comput.* 4(1), 67–95 (1986)