Explaining the Incorrect Temporal Events During Business Process Monitoring by means of Compliance Rules and Model-based Diagnosis

María Teresa Gómez-López and Rafael M. Gasca
Department of Computer Languages and Systems
University of Seville
Seville, Spain
{maytegomez, gasca}@us.es

Stefanie Rinderle-Ma
University of Vienna
Faculty of Computer Science
stefanie.rinderle-ma@univie.ac.at

***Abstract*—Sometimes the business process model is not known completely, but a set of compliance rules can be used to describe the ordering and temporal relations between activities, incompatibilities, and existence dependencies in the process. The analysis of these compliance rules and the temporal events thrown during the execution of an instance, can be used to detect and diagnose a process behaviour that does not satisfy the expected behaviour. We propose to combine model-based diagnosis and constraint programming for the compliance violation analysis. This combination facilitates the diagnosis of discrepancies between the compliance rules and the events that the process generates as well as enables us to propose correct event time intervals to satisfy the compliance rules.**

***Keywords*-Business Process Compliance, Compliance Rules, Event Analysis, Constraint Programming, Model-based Diagnosis**

I. Introduction

A business process consists of a set of activities that are performed in coordination within an organizational and technical environment [1]. The base of Business Process Management Systems (BPMS) is the explicit representation of business processes with their activities and the execution constraints between them. Sometimes, the description of the model is not known completely, but some parts of the behaviour of the process can be known and represented by means of compliance rules in a declarative way. These compliance rules describe the ordering and temporal relations between activities as well as incompatibilities and existence dependencies in a business process [2], [3], [4], [5]. In case the process model is not available at all or only partly known, compliance of running process instances has to be monitored during runtime [6]. The monitoring of the process can be carried out by means of the temporal events (henceforth events) that describe the execution of the activities of the process. In particular, if the activities are performed by persons or software not integrated in a BPMS that assures a correct order execution, it is possible that the activities are not executed according to the compliance rules. Previous works focused on the detection of compliance violations by monitoring [6]. Our proposal is centered not only in the detection and diagnosis, but also in the proposal of the correct intervals where the events should have been executed. Thereby, our proposal assumes that all the compliance rules are correct and the set of events analyzed represents all the activities executed in a period of time. A compliance rule violation, i.e., a fault in the process execution, is caused by so called incorrect events. By using the term incorrect event we do not imply that the event itself is incorrect, i.e., the associated activity execution, but the time instant when the event occurred was not in accordance with the imposed compliance rules.

Our proposal uses model-based diagnosis theory, that permits to discover the event responsible of a malfunction comparing the model that describes the system (expected behaviour) with the observational model (observed behaviour). Classic model-based diagnosis needs to be adapted to diagnose compliance rules for multiple process instances, since the same data can be involved in different instances at the same time. Also, it is necessary to establish the part of the model that can explain the incorrect behaviour, and the associated events in the compliance rule scenario. Examples of compliance rules that can be used in a diagnosis process are:

- c_1: If activity *A* is executed followed by an execution of activity *B*, activity *C* must be executed eventually.
- c_2: If activity *B* is not executed, activity *C* must be executed.
- c_3: If activity *C* is executed, activity *D* must be executed eventually.
- c_4: Every activity can only be executed once in each process instance.

Consider the following sequence of events monitored as observational model:
$\{$`Start_Process`, $\texttt{Event}_a$, $\texttt{Event}_d$, $\texttt{Event}_b$, $\texttt{Event}_c\}$.
Compliance rule c_3 (in connection with c_4) is violated, since $\texttt{Event}_d$ was not executed after $\texttt{Event}_c$ but before. Although this compliance violation can only be detected after $\texttt{Event}_c$ has occurred, this instant is not the root

cause for the violation, but the instant of $\texttt{Event}_d$ that has occurred before. Our proposal enables the evaluation of the actual culprit of the malfunction, finding out the minimum modification that the instance must suffer to be in accordance with the compliance rules again.

Moreover, by using model-based diagnosis, it is also possible to determine non-compliances even before the rules are violated or activated. Take the following trace of events: $\{\texttt{Start\_Process, Event}_a\texttt{, Event}_d\}$
Although neither compliance rule has been violated yet[1], we can assure that not any occurrence of $\texttt{Event}_c$ or $\texttt{Event}_d$ exists that does not lead to a violation of compliance rules c_1 to c_4. If $\texttt{Event}_b$ is executed or not, $\texttt{Event}_c$ has to be executed before. This is never possible since $\texttt{Event}_d$ has been already executed and cannot be executed again (c_4). The model-based diagnosis process proposed in this paper informs about the responsible of the malfunction, and how to solve this malfunction, for the example: execute *C* before *D*. This constitutes a novel pro-active detection and follow-up treatment of compliance violations before they actually occur.

Altogether, we have centered our proposal on three contributions:

- Detecting violations of compliance rules during runtime using events for multiple instances, enabling proactive treatment of future violations.
- Determining compliance-violating events. It enables fine-grained feedback and recovery to know not only the violated rules, but also the event or events that have provoked the fault.
- Determining the correct time interval where the responsible events for compliance violations should have occurred.

As it is possible that multiple instances of the same business process model can be involved in the same diagnosis process, we will also provide the necessary guidelines to enable the diagnosis of events for multiple instances collected in the same monitoring process.

The paper is organized as follows: Section II presents the compliance rule language based on graphs that we use to represent the activities temporal order relation. Also an example has been included in that section to facilitate the understanding. Section II-B analyses the necessity to determine the trace of events when multiple instances participate in the diagnosis. Section III describes how to model the compliance rules by means of numerical constraints. Section IV explains how model and compute the problem using model-based diagnosis theory to obtain the possible correct time intervals to satisfy the rules. In that section, the Constraint Programming paradigm is introduced, since it allows us to compute the diagnosis automatically depending on the observations. Section V analyses the main papers related to this work. Finally, conclusions and future work are also included in the document.

[1]and there is no conflict between compliance rules c_1 to c_4

II. Preliminaries

In this section we provide necessary background information. In particular, we introduce event and data context as necessary means for compliance analysis in a multiple instance setting.

A. Compliance Rule Description

Many languages have been proposed to describe compliance rules. Some of them will be analyzed in Section V. In order to facilitate the correct description of the compliance rules, it is necessary to take into account the complexity of the specification language. It must neither become an obstacle for constraint specification nor for the validation of processes against constraints. Thus, it is important to find an appropriate balance between expressiveness, formal foundation, and efficient analysis. The constraint specification language used in this paper is based on Compliance Rule Graphs (CRGs) [4]. We opted for CRGs since they provide a visual representation, enable the representation of common compliance rule patterns, and are particularly suited for compliance monitoring [6].

Basically, a CRG enables the graphical specification of a compliance rule c over a set of process activity types A. Specifically, a node of the CRG maps to an activity type *at* $\in A$ which could be presented in the processes which c is imposed on. In Figure 1, for example, the CRG reflecting compliance rule $c1$ consists of nodes that map to activity types `Payment run`, `Transfer to bank`, and `Check bank statement`. A CRG always follows the structuring into to node sets reflecting the *antecedent* and *consequence* parts of the rule. The antecedent nodes of the CRG represent all activity types that trigger the associated compliance rule. The consequence nodes reflect the necessary consequences a compliance rule is imposing in case the compliance rule is met. For both, antecedent and consequence nodes, a further distinction is made into *occurrence* and *absence* nodes. Occurrence nodes reflect the presence of an activity type in the underlying process. Absence of certain activity types is expressed by absence nodes respectively. In Figure 1, `Payment run` is an antecedent occurrence node meaning that $c1$ is triggered if an activity of type `Payment run` is present in a process. `Transfer to bank` and `Check bank statement` are consequence occurrence nodes reflecting that in case a `Payment run` has taken place, both of them have to happen afterwards. As it can be seen from this example, some order can be imposed on the nodes of the same type, i.e., antecedent and consequence. For example, `Transfer to bank` and `Check bank statement` have not only to be executed after `Payment run` has occurred regarding $c1$, but also in this given order. Edges between antecedent and consequence nodes describe implication relations. On top of this control flow related structures, data extensions for CRGs exist [7].

In order to introduce how the rules can be described by means of CRGs, and explain how the model can be diagnosed, an example about bank transfers presented in [8] is used. Figure 1 depicts the example by means of the CRGs, whose description is:

- c_1: Conducting a payment run creates a payment list containing multiple items that must be transferred to the bank. Then, the bank statement must be checked for payment of the corresponding items.
- c_2: For payment runs with amount beyond 10,000 €, the payment list has to be signed before being transferred to the bank and has to be filed afterwards for later audits.
- c_3: When payment of an open item is confirmed in the bank statement, the item has to be marked as cleared eventually.
- c_4: Once marked as cleared, the item must not be put on any payment list.

B. Diagnosis of Events for Multiple Instances: Data and Event Contexts

In this paper, we exploit the graphical notation of CRGs and translate them into a Constraint Satisfaction Problem later on.[2] Anyway, the abstraction to the process will be the event trace. By an event we refer to the observable execution of an activity in a business process during monitoring. The event is defined for the tuple ⟨Time, Event Context⟩, being Time $\in \mathbb{R}^{\geq 0}$. The Event Context becomes relevant for analyzing compliance in case of multiple instances and will be defined in the following.

Supposing that an event represents the execution of an activity in an instant, it is possible that the same activity would be executed several times in the same instance, or for several instances. For carrying out the diagnosis it is crucial to distinguish between both cases. For example, if two events e_1 and e_2 represent two executions of activity *Payment run* for different instances, and another event e_3 occurs due to execution of activity *Transfer to bank*, it would be necessary to know if the event e_3 is related to the event e_1 or e_2. This problem is related to the question which event model is uses. If all events are equipped with unique references to the instances they represent, the relation between events is more easy to determine. However, the existence of such unique references cannot be assumed for all application domains, particularly, if the events stem from heterogeneous sources [8]. Existing event models such as XES [9] typically equip events with time information and attributes. However, the definition of the event context to be used in associating events for later diagnosis has not explicitly been addressed. In these cases, the association between events has to be determined based on different information that enables establishing an association. A similar concept has been provided by correlation in BPEL [10]. Adopting the idea of correlating events to instances via data, we will exploit context, i.e., **Data Context** or **Event Context** that is shared between the events reflecting the underlying activity executions. In the following, we present the context model depicted in Figure 2: a compliance rule refers to a set of activities. As these activities are connected to process data, the compliance rule can be associated with a data context, that is reflected by the events occurring in the run of the activity executions. The latter is captured within the event context, reflecting all data contexts an event might refer to.

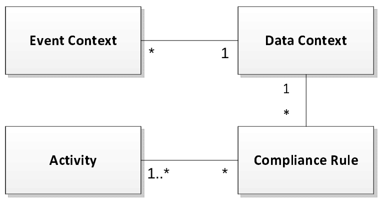


Figure 2. Conceptual model of Compliance Rules, Activities, Data Contexts and Event Contexts

The data context of a compliance rule c is described by a set of pairs ⟨Name, type⟩. In Figure 1, the data context associated to compliance rules c_1 or c_2 is {Payment list: string}, and for c_3 or c_4 the data context associated to each of them is {Payment list: string, Item: integer}. Although each compliance rule is only defined for one data context, and an activity can participate in more than one compliance rule, it is not possible that an activity is involved in two compliance rules defined for different data contexts. In Figure 3 the relation between data contexts (Payment and {Payment List, Item}), compliance rules and activities for the example is shown.

The event context of an event e is defined as the set of specific values of each event *e* for the data context of the activity that the event represents. Then, each event context is described by the triple ⟨activity, instantiation of the data context, list of information associated with the event⟩, that represents the activity that was executed, the specific instantiation for the data context associated to the activity of the type specified in the data context, and optionally a data value if it is necessary for the compliance rules (as in the compliance rule c_2 of Figure 1).

An example of reception of events, where the time is represented by means of a number of seconds after a time reference, is:

e_1 = ⟨1584, ⟨Payment run, {Payment list: A}, amount = 60.000⟩⟩
e_2 = ⟨2145, ⟨Transfer to bank, {Payment list: A}⟩⟩
e_3 = ⟨2589, ⟨Check bank statement, {Payment list: A}⟩⟩
e_4 = ⟨3256, ⟨File payment list, {Payment list: A}⟩⟩
e_5 = ⟨3267, ⟨Payment confirmed, {Payment list: A, item: 1}⟩⟩

[2]The formal semantics of CRGs is based on First Order Logic. The abstraction of the processes is provided by using event traces.

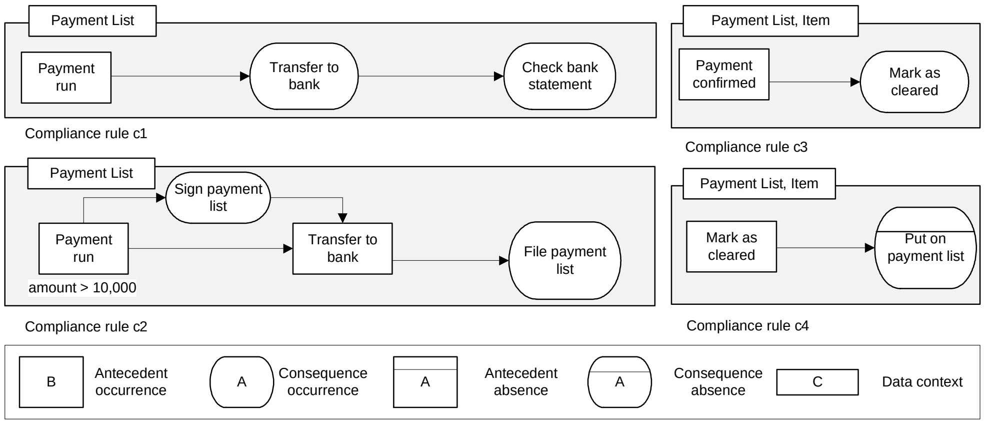


Figure 1. CRGs for the example

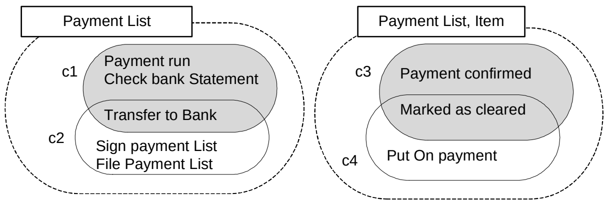


Figure 3. Relation between Data Contexts, Compliance Rules and Activities

e_6 = ⟨3589, ⟨Payment confirmed, {Payment list: A, item: 2}⟩⟩
e_7 = ⟨4435, ⟨Mark as cleared, {Payment list: A, item: 2}⟩⟩
e_8 = ⟨4431, ⟨Payment confirmed, {Payment list: A, item: 3}⟩⟩
e_9 = ⟨5214, ⟨Mark as cleared, {Payment list: A, item: 3}⟩⟩
e_{10} = ⟨5512, ⟨Put on payment list, {Payment list: A, item: 3}⟩⟩

For example, if an event e_j has an event context ⟨Payment confirmed, {Payment list: A, Item: 2}⟩, it expresses that the activity *Payment confirmed* has been executed with the specific data context (A, 2).

Finally, compliance rules as well as process activities can be clustered with respect to their data context (see Definition 1), i.e., the data they share and operate on. Based on such clusters it becomes possible to monitor and diagnose the compliance of multiple instances during runtime. Another approach to cluster compliance rules was proposed in [11], but there the clustering was done based on the processes a rule refers to not the data context.

Definition 1 (Activity and Compliance Rule Cluster (dc: Data Context))**.** *Let C be a set of compliance rules and $\mathcal{A}$ a set of activities. Then we define compliance rule cluster C_{dc} and activity cluster A_{dc} based on data context dc as follows:*

$$C_{dc} := \{c \in C \mid DataCtxt(c) = dc\}$$
$$A_{dc} := \{a \in A \mid c\ defined\ over\ A \wedge DataCtxt(c) = dc\}$$

It is not possible that two events of an activity during a monitoring cycle have the same identifier for the event context. For example, it is not possible that two events of the activity *Payment run* have the identifier of context {Payment list: A} in the same cycle of monitoring, since it is necessary to establish the trace of the events during the execution.

III. Describing Compliance rules by means of Numerical Constraints

In order to carry out the model-based diagnosis, the compliance rules and the events monitored need to be included in the model to solve. As we aim at finding out the possible correct time intervals for a malfunction, it is necessary to use a model that permits to include numerical aspects, such as numerical constraints to describe the time instant of the event executions, not only the temporal order relation between them.

A. Modeling events by means of Numerical Constraints

As it is introduced in the previous section, an event is described by means of an instant, and an event context. It implies that an observational model is described by a sequence of events, ordered in function of the instant when they were thrown σ = <e_1, ..., e_n> [12], and where this trace can involve different instances of the same process model. In this paper we assume that events are only represented by means of an occurrence in an instant, not with a duration.

To diagnose a compliance rule violation in a business process, it is necessary to model the executed events and the events that can be executed in the future as well, to determine possible non-compliances according to the compliance rules. Obviously, it is not possible to know the events that will be executed in the future, but we have some information derived from the use of the compliance rules that describe the model. For example, analysing the compliance rule $c4$ of the example, if an event related to the activity *Mark as* *cleared* is executed, it is possible that an event for the activity *Put on payment list* will be executed in the future, although it is described as an incorrect behaviour of the process. In order to model the scenario to diagnose, the executed events and the non-executed events must be included as well, since they can be related by means of the different compliance rules that govern the process behaviour. For example, if activity $\mathtt{A}$ must be executed after activity $\mathtt{B}$, and an event about $\mathtt{A}$ was executed in the instant t_z, to satisfy the compliance rule, an event of B has to be executed after the instant t_z and before the instance ends. As the event included in the model can represent events thrown in the past, or the events that can be executed in the future, we propose to include in the model-based diagnosis for each event (executed or not) a new variable associated to the timestamp, where the pair of variables $\langle$Executed, 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*. Depending on the instant when an event is executed, both parameters (*Executed* and *Time*) can take different values. It has to satisfy the following rules depending on the execution of each event:

- If the event *e* has been executed: *Executed* $=$ *true* $\wedge$ Initial Time $\leq$ *Time* $\leq$ current Time ($event_i$ in Figure 4)
- If the event *e* has not been executed but will be executed in the future: *Executed* $=$ *true* $\wedge$ *Time* $>$ current Time ($event_j$ in Figure 4)
- It the event *e* has not been executed and will not be executed in the future: *Executed* $=$ *true* $\wedge$ *Time* $= -1$

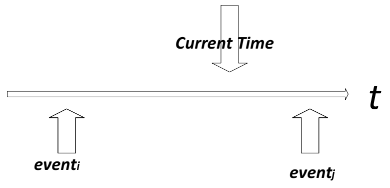


Figure 4. Possible execution of events

B. Modeling Compliance Rule Graphs by means of Numerical Constraints

In order to infer if an activity has been executed in an incorrect instant, according to a set of events, we propose to transform each CRG into a numerical constraint to be combined with the model of execution of events explained in the previous subsection. The inclusion of numerical aspects into compliance rule validation process enables proactive treatment of future violations. Each edge involved in the CRG is transformed into a numerical constraint to represent the time and temporal order execution of the activities. Following the possible patterns of relation that can exist for the components of a CRG, Figure 5 shows the transformation of each combination of nodes in a logical formula that will be included in the numerical model to diagnose the business process execution. In Figure 5, the execution of an activity $\mathtt{A}$ and the instant when it is executed are represented by means of the variables $\mathtt{A}_{Ex}$ and $\mathtt{A}_T$ respectively. The patterns put the constraints of antecedents before the implication operator '$\rightarrow$', and the constraints of the consequences after it. The temporal order is represented related the variables $\mathtt{A}_T$ with the '$<$' operator, and the occurrences and absences by means of the Boolean variables $\mathtt{A}_{Ex}$ for the different activities.

For the example of Section II, the constraints that describe the compliance rules according to the CRGs are:

- c_1: $\mathtt{PaymentRun}_{Ex} \rightarrow (\mathtt{TransferToBank}_{Ex} \wedge \mathtt{PaymentRun}_T < \mathtt{TransferToBank}_T) \wedge (\mathtt{CheckBankstatement}_{Ex} \wedge \mathtt{TransferToBank}_T < \mathtt{CheckBankstatement}_T)$
- c_2: $(\mathtt{PaymentRun}_{Ex} \wedge \mathtt{amount} > 10{,}000 \wedge \mathtt{TransferToBank}_{Ex} \wedge \mathtt{PaymentRun}_T < \mathtt{TransferToBank}_T) \rightarrow (\mathtt{SignPaymentList}_{Ex} \wedge \mathtt{PaymentRun}_T < \mathtt{SignPaymentList}_T \wedge \mathtt{SignPaymentList}_T < \mathtt{TransferToBank}_T) \wedge (\mathtt{FilePaymentList}_{Ex} \wedge \mathtt{TransferToBank}_T < \mathtt{FilePaymentList}_T)$
- c_3: $\mathtt{PaymentConfirmed}_{Ex} \rightarrow \mathtt{MarkAsCleared}_{Ex} \wedge \mathtt{PaymentConfirmed}_T < \mathtt{MarkAsCleared}_T$
- c_4: $\mathtt{MarkAsCleared}_{Ex} \rightarrow \neg(\mathtt{PutOnPaymentList}_{Ex} \wedge \mathtt{PutOnPaymentList}_T > \mathtt{MarkAsCleared}_T)$

These constraints are close to the model necessary to diagnose, but they are not exactly the same, since multiple instances have to be taken into account. The next sections present how to create the different diagnosis models depending on the observed events for several instances.

IV. Creating Diagnosis Models for Several Instances

Model-based diagnosis analysis is used to ascertain whether the behaviour of a system is correct or not, and who is the responsible of the malfunction. This identification is carried out by comparing the expected behaviour (the model) with the observed behaviour (the observational model). As in this paper we deal with the diagnosis of multiple instances of a process, where an activity can be executed more than once in each instance, it is necessary to introduce two types of diagnosis models, the static model that describes the compliance rules for any instance (as explained in Section III), and the dynamic models formed by the compliance rules for each instance found in the observational model, thereby related to the data contexts and event contexts.

For this reason, we extend the architecture presented in [8], including the necessary modules to create dynamic models according to the observational model to diagnose the incorrect events, and propose the event time execution intervals that make all the compliance rules satisfiable (Figure 6). Each dynamic model is automatically created and

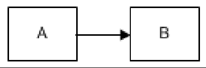
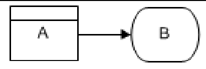
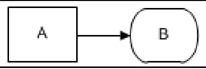
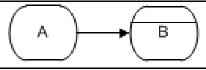
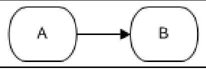
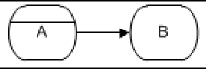
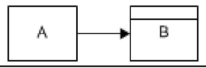
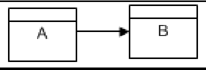
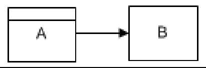
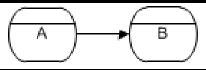
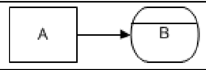
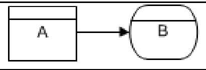
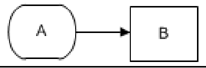
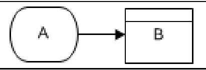
Template	Logical Formula	Template	Logical Formula
	$A_{Ex} \wedge B_{Ex} \wedge A_T < B_T$		¬ AEx à BEx
	$A_{Ex} \wedge B_{Ex} \rightarrow A_T < B_T$		$\rightarrow A_{Ex} \wedge \neg(B_{Ex} \wedge A_T<B_T)$
	$\rightarrow A_{Ex} \wedge B_{Ex} \wedge A_T < B_T$		$\rightarrow B_{Ex} \wedge \neg(A_{Ex} \wedge A_T<B_T)$
	$A_{Ex} \wedge \neg(B_{Ex} \wedge A_T < B_T)$		$\neg A_{Ex} \wedge \neg B_{Ex}$
	$B_{Ex} \wedge \neg(A_{Ex} \wedge A_T < B_T)$		$\rightarrow \neg A_{Ex} \wedge B_{Ex}$
	$A_{Ex} \rightarrow \neg(B_{Ex} \wedge A_T < B_T)$		$\neg A_{Ex} \rightarrow \neg B_{Ex}$
	$B_{Ex} \rightarrow (A_{Ex} \wedge A_T < B_T)$		$\neg B_{Ex} \rightarrow A_{Ex}$

Figure 5. Patterns of transformation from CRGs to numerical constraints

solved using constraint programming paradigm as explained in the following subsections.

A. Creating the Diagnosis Dynamic Models

As it was commented in Section II-B, our proposal allows the diagnosis of several instances included in the same observational model. It is related to the definition data contexts, that permits the differentiation of the different instances in a business process execution. Depending on the data contexts that exist for a set of events observed, the dynamic diagnosis models will be different. For each set of events, related between them to belong to the same data context, will be necessary to define compliance rules represented by means of numerical constraints that describe their relations. The compliance rules that describe the activities order relation will be used as a pattern for the different dynamic models that will be created in function of the observational model. In order to clarify the difference between static, observational and dynamic models, the following definitions are introduced:

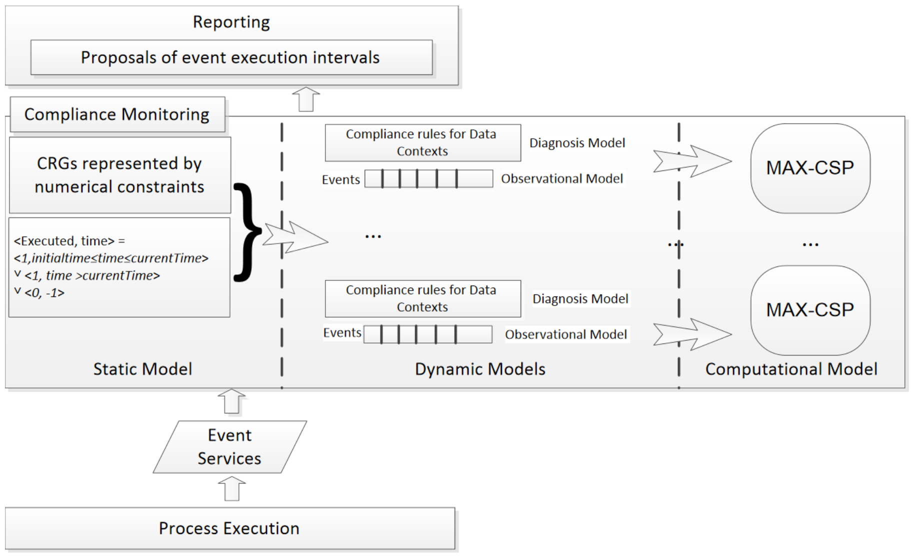
Definition 2 (Static Diagnosis Model (SDM))**.** *It is formed by:*

- *The activities $\{a_1, \ldots, a_n\}$ of the business process.*
- *The compliance rules represented by numerical constraints $\{c_1, \ldots, c_m\}$ following the patterns shown in Figure 5.*
- *The data contexts $\{dc_1, \ldots, dc_f\}$.*
- *The activity clusters for each data context following Definition 1: $A_{dc}(dc_1) \rightarrow \{a_i, \ldots, a_j\}$, $A_{dc}(dc_2) \rightarrow \ldots$*
- *The compliance rule clusters for each data context following Definition 1: $C_{dc}(dc_1) \rightarrow \{c_i, \ldots, c_j\}$, $C_{dc}(dc_2) \rightarrow \ldots$*

Definition 3 (Observational Model (OM))**.** *The events $\{e_1, \ldots, e_p\}$ that make visible the execution of the activities out of the process for the Compliance Monitoring layer (Figure 6). Each event is associated to one and only one activity, although an activity can be represented by several events. The observational model is also composed of the current time and the initial time for each instance.*

Definition 4 (Dynamic Diagnosis Model (DDM))**.** *It is created according to the SDM and the OM, then for the same SDM different DDM can be defined depending on the events involved in the cycle of monitoring. The DDM is formed by:*

- *The event contexts $\{ec_1, \ldots, ec_d\}$ for the OM that are described by a tuple $\langle dc, value\rangle$ where dc represents the data context and value represents the specific value of the data context. For the example of Section II, the event contexts obtained are: $\{\langle$Payment list, A$\rangle$, $\langle$(Payment list, Item), (A, 1)$\rangle$, $\langle$(Payment list, Item), (A, 2)$\rangle$, $\langle$(Payment list, Item), (A, 3)$\rangle\}$.*
- *Being $\{v_1, \ldots, v_n\}$ all the different values of a data context dc for an OM, for each value $v_i \in \{v_1, \ldots, v_n\}$, a set of compliance rules associated to the data context dc (C_{dc} (dc)) is created $\{{c_1}^{v_i}, {c_2}^{v_i}, \ldots\}$. The variables that represent the execution of the activities and the timestamp, in the patterns shown in Figure 5, will be adjusted for the different values of the data contexts. Then, the variables of the numerical constraints will be A_{Ex}^{value} and A_T^{value} that represent the execution of the activity A in an event context with a specific value.*
- *A_{Ex}^{value} and A_T^{value} are also related by means of the constraint: ($A_{Ex}^{value} \wedge InitialTime \leq A_T^{value} \leq currentTime$) $\vee$ ($A_{Ex}^{value} \wedge currentTime < A_T^{value}$) $\vee$ ($\neg A_{Ex}^{value} \wedge currentTime = -1$).*



Reporting
Proposals of event execution intervals
Compliance Monitoring
CRGs represented by numerical constraints
<Executed, time> = <1,initialtime≤time≤currentTime> ∨ <1, time >currentTime> ∨ <0, -1>
Static Model
Compliance rules for Data Contexts
Diagnosis Model
Events
Observational Model
...
Dynamic Models
MAX-CSP
Computational Model
Event Services
Process Execution

Figure 6. Business Rule Monitoring Architecture

The constraints of the compliance rules for the DDM of the example of Section II for the events $\{e_1, \ldots, e_{10}\}$ are:

- Compliance rules for the Data Context *Payment list* for the value *A*:
  - c_1^A: $\text{PaymentRun}_{Ex}^A \rightarrow (\text{TransferToBank}_{Ex}^A \wedge \text{PaymentRun}_T^A < \text{TransferToBank}_T^A) \wedge (\text{CheckBankstatement}_{Ex}^A \wedge \text{TransferToBank}_T^A < \text{CheckBankstatement}_T^A)$
  - c_2^A: $(\text{PaymentRun}_{Ex}^A \wedge \text{amount} > 10{,}000 \wedge \text{TransferToBank}_{Ex}^A \wedge \text{PaymentRun}_T^A < \text{TransferToBank}_T^A) \rightarrow (\text{SignPaymentList}_{Ex}^A \wedge \text{PaymentRun}_T^A < \text{SignPaymentList}_T^A \wedge \text{SignPaymentList}_T^A < \text{TransferToBank}_T^A) \wedge (\text{FilePaymentList}_{Ex}^A \wedge \text{TransferToBank}_T^A < \text{FilePaymentList}_T^A)$
- Compliance rules for the Data Context (*Payment list, Item*) for the value *(A, 1)*:
  - $c_3^{A,1}$: $\text{PaymentConfirmed}_{Ex}^{A,1} \rightarrow \text{MarkAsCleared}_{Ex}^{A,1} \wedge \text{PaymentConfirmed}_T^{A,1} < \text{MarkAsCleared}_T^{A,1}$
  - $c_4^{A,1}$: $\text{MarkAsCleared}_{Ex}^{A,1} \rightarrow \neg(\text{PutOnPaymentList}_{Ex}^{A,1} \wedge \text{PutOnPaymentList}_T^{A,1} > \text{MarkAsCleared}_T^{A,1})$
- Compliance rules for the Data Context (*Payment list, Item*) for the value *(A, 2)*:
  - $c_3^{A,2}$: $\text{PaymentConfirmed}_{Ex}^{A,2} \rightarrow \text{MarkAsCleared}_{Ex}^{A,2} \wedge \text{PaymentConfirmed}_T^{A,2} < \text{MarkAsCleared}_T^{A,2}$
  - $c_4^{A,2}$: ...
- Compliance rules for the Data Context (*Payment list, Item*) for the value *(A, 3)*:
  - $c_3^{A,3}$: ...

B. Solving Diagnosis Models by means of Constraint Programming

The DDM and the OM determine the necessary compliance rules to describe the instance-executions, but this model needs to be transformed into a computable model to be diagnosed. For this reason, we propose the use of Constraint Programming (CP) paradigm [13], building and solving Constraint Satisfaction Problems (CSP) automatically from the DDM. The use of CP permits the combination of numerical and Boolean variables to represent the model and deduce the incorrect events and the possible correct time intervals where they should have been executed.

The 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 $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \ldots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \ldots, C_m\}$ is a set of constraints. 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 [14]. The consistency techniques remove non-compliance values from the domains of the variables during or before the search. Several local consistency and optimization techniques have been proposed as ways of improving the efficiency of search algorithms, being especially optimized for linear problem. CP is an area in continuous evolution, with important commercial tools and with an active research.

In order to compute the DDM, we propose to model a CSP where:

- *X* is formed by all the variables A_{Ex}^{value} and A_T^{value} that

represent the execution of each activity (execution and time) that are necessary to represent the instances.
- *D* is defined as Boolean for the variables A_{Ex}^{value}, and Integer for the variables A_{T}^{value}. A_{T}^{value} is represented by means of the absolute number of seconds or milliseconds (depending on the necessary granularity in each problem) that have elapsed since midnight, January 1, 1970, the typical time reference used in the libraries of some programming languages (for example *System.currentTimeMillis* in Java^{TM}).
- *C* is the set of compliance rules represented by means of constraints derived from the OM as explained in Definition 4. These compliance rules are defined over the variables A_{Ex}^{value} and A_{T}^{value}. Moreover, it is necessary to assign the values of the observed events to the variables A_{Ex}^{value}, A_{T}^{value}, initial time and current time.

If there is a tuple of values for the variables *X* in the domain *D*, where all the constraints *C* are satisfiable, the CSP solver will return a tuple with the possible values of *X*, then we can assure that the compliance rules are satisfiable for the OM. The problem is if there is no a tuple of values where the set *C* is satisfiable, we would not have information about the malfunction, only that, at least, one event is incorrect. We propose to find out the minimal non-compliance subset of events that explain the malfunction. This is equivalent to maximize the number of events that were thrown in a correct instant. In order to understand how we can model the CSP to obtain that, two notions need to be introduced: Reified Constraints and Constraint Optimization Problems.

A reified constraint consists of a constraint associated to a Boolean variable which denotes its truth value. The diagnosis that we propose is centered in the detection of incorrect instant of event execution, then it is possible that the assignment of an instant to an event (A_{T}^{value}) will be incorrect. For this reason, in the CSP we do not associate mandatorily to each A_{T}^{value} the timestamp where it was executed, then we associate a Boolean variable to each assignment of a specific value to each variable A_{T}^{value}. Being the objective to maximize the number of events that occurs when the Compliance Monitoring layer describes. As the objective is to maximize the number of reified variables that are true, then an objective function is necessary. When an objective function *f* has to be optimized (maximized or minimized), then a Constraint Optimization Problem (COP) is used, which is a CSP and an objective function *f*. A COP modelled to find the assignment of reified constraints is called maximal Constraint Satisfaction Problem (Max-CSP). A Max-CSP consists of finding out a total assignment which satisfies the maximum number of constraints. Max-CSP is an NP-hard problem and generally is more complex to solve than a CSP. The basic complete method for solving this problem was designed by Wallace [15]. Max-CSPs have already been used in model-based diagnosis [16], although never for business processes. Different algorithms have been proposed in order to improve the obtaining of all minimal unsatisfiable subsets using notions of independence of constraints and incremental constraint solvers [17] and structural analysis [18].

For our model-based diagnosis, the objective function to maximize is $\{\mathrm{rf}_1 + \ldots + \mathrm{rf}_n\}$, where each $\mathrm{rf}_i \in \{\mathrm{rf}_1, \ldots, \mathrm{rf}_n\}$ represents a reified constraint for each event of the OM. The CSP solver finds out the minimum combination of incorrect events that explain the malfunction and the correct time intervals for incorrect events. The COP obtained after the transformation of the DDM for the previous example is shown in Table I.

C. Results for the example

For the example, if the diagnosis process is executed after the event e_{10} is thrown, the set of intervals obtained by means of our proposal is:

- To execute *Check bank statement* activity of *Payment list* A between [2146..instance ends]
- To execute *Sign payment list* of *Payment list* A between [1585..2144]
- To execute *File payment list* of *Payment list* A between [2146..instance ends]
- To execute *Marked as cleared* of {*Payment list* A, *Item* 1} between [3268..instance ends]

V. Related Work

There are many proposals that use the compliance rules and the monitoring of events to verify the correctness of a business process instance [19], [5], [8]. The different solutions depend on the compliance rule language used to describe the model. In order to describe the compliance rules, 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 modeling 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 [20]. The differences between declarative process languages are centered in the different perception of what is an state. Some examples are the case handling paradigm [21], the constraint specification framework [22], the ConDec language [23], the PENELOPE language [24], or EM-BrA^{2}CE [25]. These languages use different knowledge representation paradigms, that enable different types of compliance rule verification. For instance, the ConDec 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

```
//Variables and Domains:
  PaymentRun$^{A}_{Ex}$, PaymentConfirmed$^{A,1}_{Ex}$, PaymentConfirmed$^{A,2}_{Ex}$, ...: Boolean
  PaymentRun$^{A}_{t}$, PaymentConfirmed$^{A,1}_{t}$, PaymentConfirmed$^{A,2}_{t}$, ...: Integer
  ...
  rf$_1$, ..., rf$_{10}$: Boolean //one for each event
//Initialization of variables
  currentTime = 5800
  initialTime = 1200
//Compliance rules for each different value of data contexts
  c$_1^{A}$: PaymentRun$^{A}_{Ex}$ $\rightarrow$ TransferToBank$^{A}_{Ex}$ ...
  ...
  c$_3^{A,1}$: PaymentConfirmed$^{A,1}_{Ex}$ $\rightarrow$ MarkAsCleared$^{A,1}_{Ex}$ ...
  ...
  c$_3^{A,2}$: PaymentConfirmed$^{A,2}_{Ex}$ $\rightarrow$ MarkAsCleared$^{A,2}_{Ex}$ ...
  ...
//Reified constraints for each event for the OM
  rf$_1$ = (PaymentRun$^{A}_{T}$ = 1584)
  rf$_2$ = (TransferToBank$^{A}_{T}$ = 2145)
  ...
  rf$_{10}$ = (CheckBankDetails$^{A,3}_{T}$ = 5512)
//Constraints to describe time relation
  (PaymentRun$^{A}_{Ex}$ $\wedge$ InitialTime $\leq$ PaymentRun$^{A}_{T}$ $\leq$ currentTime) $\vee$ ...)
  (MarkedAsCleared$^{A}_{Ex}$ $\wedge$ InitialTime $\leq$ MarkedAsCleared$^{A}_{T}$ $\leq$ currentTime) $\vee$ ...)
  ...
  (PaymentConfirmed$^{A,1}_{Ex}$ $\wedge$ InitialTime $\leq$ PaymentConfirmed$^{A,1}_{T}$ $\leq$ currentTime) $\vee$ ...)
  ...
  (PaymentConfirmed$^{A,2}_{Ex}$ $\wedge$ InitialTime $\leq$ PaymentConfirmed$^{A,2}_{T}$ $\leq$ currentTime) $\vee$ ...)
  ...
//Objective Function
  Maximize (rf$_1$ + ... + rf$_{10}$)
```

Table I
EXAMPLE OF CONSTRAINT OPTIMIZATION PROBLEM

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 automatons does not allow to infer the correct time intervals even before the compliance rules are activated. 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 diagnosis process as in [26], but with the difference that in this case a declarative language in used Instead of an imperative language. The Event Calculus [27] 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 malfunction, inferring errors in the future.

To the best of our knowledge there are no proposals that provide the correct time intervals of the events monitored in a business process for several instances, even before the compliance rules are violated.

VI. CONCLUSIONS AND FUTURE WORKS

We have presented a framework to diagnose, even before the compliance rules are violated, the non-conformity of the set of temporal events in accordance to the compliance rules that describe the behaviour of a business process. The diagnosis method that we propose, creates the necessary dynamic models depending on the events analysed and the possible events that could be executed in the future. It permits the diagnosis of errors even before they occur. The diagnosis and prognosis not only determine the incorrect event, but find out the possible time intervals where the events should have been executed to satisfy all the compliance rules. It can be used at run time during instance-executions, or as a post-mortem process to prevent failures in future instances. As future work we plan to use this framework to detect possible patterns of malfunctions, for example, that an activity is frequently executed later. Also we think that can be interesting to apply our proposal to other declarative languages that permit the description of other types of relations between activities.

ACKNOWLEDGMENT

The work presented in this paper has been partly funded by the Austrian Science Fund (FWF):I743, the Ministry of Science and

Technology of Spain (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

References

[1] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] A. Awad, M. Weidlich, and M. Weske, "Visually specifying compliance rules and explaining their violations for business processes," *J. Vis. Lang. Comput.*, vol. 22, no. 1, pp. 30–55, 2011.

[3] S. Sadiq, G. Governatori, and K. Namiri, "Modeling control objectives for business process compliance," in *Proceedings of the 5th international conference on Business process management*, ser. BPM'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 149–164. [Online]. Available: http://dl.acm.org/citation.cfm?id=1793114.1793130

[4] L. T. Ly, S. Rinderle-Ma, and P. Dadam, "Design and verification of instantiable compliance rule graphs in process-aware information systems," in *CAiSE*, 2010, pp. 9–23.

[5] F. Maggi, M. Montali, M. Westergaard, and W. van der Aalst, "Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata," in *Proc. of BPM*, ser. LNCS. Springer-Verlag, 2011.

[6] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. van der Aalst, "A framework for the systematic comparison and evaluation of compliance monitoring approaches," in *17th Int'l EDOC Conference*, 2013, (accepted for publication).

[7] D. Knuplesch, L. T. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam, "On enabling data-aware compliance checking of business process models," in *ER*, 2010, pp. 332–346.

[8] L. T. Ly, S. Rinderle-Ma, D. Knuplesch, and P. Dadam, "Monitoring business process compliance using compliance rule graphs," in *19th International Conference on Cooperative Information Systems (CoopIS 2011)*, ser. LNCS, no. 7044. Springer, 2011, pp. 82–99.

[9] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "XES, XESame, and ProM 6," in *Information Systems Evolution - CAiSE Forum 2010*, vol. 72, 2010, pp. 60–75.

[10] S. Hinz, K. Schmidt, and C. Stahl, "Transforming bpel to petri nets," in *Business Process Management*. Springer, 2005, pp. 220–235.

[11] S. Rinderle-Ma, S. Kabicher, and L. T. Ly, "Activity-oriented clustering techniques in large process and compliance rule repositories," in *Business Process Management Workshops*. Springer, 2012, pp. 14–25.

[12] Y. Pencolé and A. Subias, "A chronicle-based diagnosability approach for discrete timed-event systems: Application to web-services," *J. UCS*, vol. 15, no. 17, pp. 3246–3272, 2009.

[13] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc., 2006.

[14] R. Dechter, *Constraint Processing (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, May 2003.

[15] R. J. Wallace, "Directed arc consistency preprocessing," in *Constraint Processing, Selected Papers*, 1995, pp. 121–137.

[16] R. Ceballos, R. M. Gasca, C. D. Valle, and M. Toro, "Max-csp approach for software diagnosis," in *IBERAMIA*, 2002, pp. 172–181.

[17] M. G. de la Banda, P. J. Stuckey, and J. Wazny, "Finding all minimal unsatisfiable subsets," in *PPDP '03: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming*. ACM Press, 2003, pp. 32–43.

[18] R. M. Gasca, C. D. Valle, M. T. G. López, and R. Ceballos, "Nmus: Structural analysis for improving the derivation of all muses in overconstrained numeric csps," in *CAEPIA*, 2007, pp. 160–169.

[19] F. M. Maggi, M. Montali, and W. M. P. van der Aalst, "An operational decision support framework for monitoring business constraints," in *Proceedings of the 15th international conference on Fundamental Approaches to Software Engineering*, ser. FASE'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 146–162. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28872-2_11

[20] I. Bider, M. Khomyakov, and E. Pushchinsky, "Logic of change: Semantics of object systems with active relations," *Autom. Softw. Eng.*, vol. 7, no. 1, pp. 9–37, 2000.

[21] W. M. van der Aalst, M. Weske, and D. Grnbauer, "Case handling: A new paradigm for business process support," *Data and Knowledge Engineering*, vol. 53, 2005.

[22] S. W. Sadiq, M. E. Orlowska, and W. Sadiq, "Specification and validation of process constraints for flexible workflows," *Inf. Syst.*, vol. 30, no. 5, pp. 349–378, 2005.

[23] M. Pesic and W. M. P. van der Aalst, "A declarative approach for flexible business processes management," in *Proceedings of the 2006 international conference on Business Process Management Workshops*, ser. BPM'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 169–180.

[24] S. Goedertier and J. Vanthienen, "Designing compliant business processes with obligations and permissions," in *Business Process Management Workshops*, 2006, pp. 5–14.

[25] S. Goedertier, R. Haesen, and J. Vanthienen, "Em-bra^{2}ce v0.1: A vocabulary and execution model for declarative business process modeling," K.U.Leuven, FETEW Research Report KBI_0728, 2007.

[26] M. T. Gómez-López, R. M. Gasca, L. Parody, and D. Borrego, "Constraint-driven approach to support input data decision-making in business process management systems," in *International Conference on Information System Development*, ser. ISD 2011. Springer, 2011, pp. 15–25.

[27] R. A. Kowalski and M. J. Sergot, "A logic-based calculus of events," *New Generation Comput.*, vol. 4, no. 1, pp. 67–95, 1986.