Event-based Real-time Decomposed Conformance Analysis

Seppe K.L.M. vanden Broucke[1], Jorge Munoz-Gama[2], Josep Carmona[2], Bart Baesens[1,3], Jan Vanthienen[1]

[1]Department of Decision Sciences and Information Management, KU Leuven, Naamsestraat 69, B-3000, Leuven, Belgium
[2]Universitat Politecnica de Catalunya, Barcelona, Spain
[3]School of Management, University of Southampton, Highfield Southampton, SO17 1BJ, United Kingdom
{seppe.vandenbroucke,bart.baesens,jan.vanthienen}@kuleuven.be
{jmunoz,jcarmona}@cs.upc.edu

Abstract. Process mining deals with the extraction of knowledge from event logs. One important task within this research field is denoted as conformance checking, which aims to diagnose deviations and discrepancies between modeled behavior and real-life, observed behavior. Conformance checking techniques still face some challenges, among which scalability, timeliness and traceability issues. In this paper, we propose a novel conformance analysis methodology to support the real-time monitoring of event-based data streams, which is shown to be more efficient than related approaches and able to localize deviations in a more fine-grained manner. Our developed approach can be directly applied in business process contexts where rapid reaction times are crucial; an exhaustive case example is provided to evidence the validity of the approach.

Keywords: real-time monitoring, process decomposition, conformance checking, conformance analysis, process mining, event logs

1 Introduction

The research field of process mining deals with the extraction of knowledge from event logs, and has situated itself in the course of the past decade between the areas of Business Process Management (BPM) and data mining. As more and more process aware information systems are implemented, an increasing amount of event-based data is being recorded, which can hence be analyzed by process mining related techniques. An important task within process mining is called *conformance checking* (or, more broadly: *conformance analysis*), which aims to diagnose deviations and discrepancies between modeled behavior and real-life, observed behavior.

Conformance checking techniques face some hard and complex challenges in the context of today's organizations. First, the increasing amount of information

systems being implemented and applied to provide operational support, drive decisions and assist managers have led to a barrage of data, which should be parsed and analyzed in a manner which is both correct and scalable by conformance checking techniques. Second, given the current turbulent economic environment, stakeholders desire more than ever the timely delivery of reports and warnings, so that conformance checking techniques should no longer be applied in a post-hoc manner, after the actual occurrence of the activities being executed. Third, such techniques should be able to quickly and correctly localize and pinpoint deviating behavior and its root causes. As process models can become very complex, one wishes to highlight misbehaving parts in a running model, together with the ability to "zoom in and out" on these elements. Many conformance checking techniques have mainly been aiming to derive a global quality "metric", denoting the global fitness or appropriateness of a process model, but without any real attention being applied towards localizing the main points of failure in an understandable manner.

In this paper, we propose a novel methodology to support real-time conformance analysis of event-based data streams, which aims to provide an answer to the challenges listed above. Our approach contributes to the current body of work in the following ways. First, we apply state of the art process model decomposition techniques [1] to split a large process model in a series of sub processes in order to gain a significant *speed-up* when verifying events. Second, by applying decomposition techniques, *localizing deviations* and volatile parts of the process models becomes more straightforward, allowing end-users to quickly gain an insight in which parts of the current model are failing or being violated. Third, by combining model decomposition techniques with a fast, event-granular replay technique, we are able to perform the conformance analysis task in a *real-time* manner, thus allowing for the monitoring of incoming events as they are being executed. This is a strong contribution compared to earlier approaches [2, 3, 4, 5, 6], where conformance checking techniques assume that a full recorded event log is available and where the actual analysis can be time-consuming.

The possible areas of application for our developed approach are manifold. In light of recent financial crises, the importance of the ability to immediately react to external shocks and unforeseen events has become more apparent than ever. Real-time monitoring, fraud detection and governance, risk and compliance (GRC) verification, and trading system failure protection all provide suitable contexts to apply our proposed technique. We apply our technique on a case example of a large bank transfer process to illustrate the validity of our contribution.

The remainder of the paper is structured as follows, Section 2 provides an overview of preliminary definitions and an overview of related work. Section 3 describes in detail our applied methodology, together with a technical description of the implemented artifact. Section 4 provides an empirical validation by describing a relevant case example and compares our technique against relevant approaches in this context. Section 5 concludes the paper and outlines opportunities for future work.

2 Preliminaries

This section provides an overview of preliminary definitions and concepts, as well as an overview of related work.

2.1 Related Work

Conformance checking techniques are devoted to quantify the quality of a process model in describing an event log. In the seminal work [7], a "fitness" metric is presented to describe the extent to which event traces can be associated with valid execution paths in the process model, and an "appropriateness" metric is proposed to asses whether the process model describes the observed behavior accurately enough. The aforementioned approach *replays* the traces of the log in the model to evaluate these metrics. One of the drawbacks of this approach is that for *undeterministic* models, the heuristics used in the replay may lead to overestimating the metrics, due to the artificial creation of superfluous tokens in the model. Several solutions have been proposed to overcome this issue. Weidlich et al. propose a system to check process model consistency based on "behavioral profiles" [8, 9]—which can be derived in a straightforward and efficient manner but with loss of some granularity regarding the exact traces which can be accepted by the model at hand. Adriansyah et al. propose an alternative approach where the concept of "alignments" are introduced in order to match an event trace with a path through the model as closely as possible [3, 5, 6].

Various *decomposition* approaches to improve process discovery and conformance checking tasks have been proposed. In [10], the notion of passages is used to decompose a process model and/or event log into smaller parts to speed-up process discovery and conformance checking. This approach has been generalized in [1] where it is shown that any event-granular process discovery and conformance checking tasks can be decomposed as long as the different process fragments (i.e. the submodels) only share uniquely-labeled activities. We apply this approach in this paper, but utilize a Refined Process Structure Tree (RPST) based decomposition method as outlined in [11, 12], as the hierarchical topological structure provided by this decomposition allows to enable additional analytical tasks not considered before, such as zooming in and out on various parts of the process model being monitored.

Our methodology also bears similarities with the fields of Complex Event Processing (CEP) [13, 14] and Business Activity Monitoring (BAM) [15, 16]. Although these methodologies also support the (near-)real-time monitoring of events, our approach differentiates itself in two ways. First, our approach stays in the general realm of process mining by starting from a process model and comparing this against a stream of incoming events which can be related to several running process instances. Second, as we apply a decomposition strategy over the given process model, this allows to immediately relate violations or discrepancies to specific areas within this model, thus improving the localization of the root-causes behind such deviations.

Our conformance analysis methodology is applicable on all process models on which event-granular semantics can be defined and which can be meaningfully decomposed into a series of submodels. We apply Petri nets throughout this paper as the representational language for prescriptive process models.

2.2 Definitions

Definition 1. ***(Petri net, Workflow net.)*** *A Petri net [17] is a triplet $PN = (P, T, F)$ with P a finite set of places and T a finite set of transitions with $P \cap T = \emptyset$. F is the set of flows $F \subseteq (P \times T) \cup (T \times P)$. A place p is an input place of a transition t iff $(p, t) \in F$; similarly, p is an output place of t iff $(t, p) \in F$. The state of a Petri net is defined by its marking $M : P \to \mathbb{N}_0$. A transition is "enabled" in a given marking whenever all of its input places contain at least one token. Firing a transition then consumes a token from each input place and produces a token in each output place.*

We also define the concepts of workflow graphs, system nets and full firing sequences.

Definition 2. ***(Workflow graph, System net, Full firing sequence.)*** *Given a Petri net $PN = (P, T, F)$, the workflow graph is defined as the structural directed graph $G = (V, E)$ with $V = P \cup T$ and $E = F$. A system net is a triplet defined over a given Petri net $SN = (PN, m_i, m_o)$ where m_i and m_o define the initial and final markings of the Petri net, respectively. $(PN, m_1)[\sigma\rangle(PN, m_2)$ denotes that a sequence of transitions $\sigma \in T$ is enabled and can be fired starting from marking m_1, resulting in marking m_2.*

Next, we provide the definition of an event log together with an event stream.

Definition 3. ***(Event log, Event stream.)*** *Let event log L be defined as a multiset of traces (process instances) with the cardinality (or size) $|L|$ denoting the total number of traces in the log, including duplicates. A trace $\sigma^L \in L$ is a finite sequence of activity labels with length $|\sigma^L|$ and with σ_i^L the activity at position i in trace σ^L. The set of activities occurring in the event log is then denoted as $A = \{\sigma_i^L | \sigma^L \in L, i = 1 \ldots |\sigma^L|\}$. For the purpose of our real-time conformance analysis methodology, we define an event stream $ES = \langle e_1, e_2, \ldots \rangle$ as a sequence (finite or infinite) of arriving events with an event $e \in ES$ expressed as a tuple $(id, act, time)$, $act : ES \to A$ a function denoting the corresponding activity label for an event and function $id : ES \to \mathbb{N}$ denoting the case identifier. It is trivial to convert an event log L to an event stream ES if a global order relation can be established over the recorded activities in the event log.*

To establish whether a given Petri net is able to correctly parse a given activity trace, a mapping between the transitions T in the Petri net and the activity alphabet A of the event log has to be established.

Definition 4. ***(Mapping, Fitting trace)*** *Given a Petri net $PN = (P, T, F)$ and an event log L with activity alphabet A, let $\mu : T \mapsto A \cup (s, b)$ be defined as a*

mapping between the transitions in the Petri net and activities in the event log, denoting the relation between a fired transition and the recorded label in the event log. Multiple transitions mapped to the same activity are denoted as duplicate transitions. Transitions can be mapped to a silent activity s (not observed in event log) and hence executed at-will whenever they are enabled. A trace σ^L "fits" a system $SN = (PN, m_i, m_o)$ when a full firing sequence $(PN, m_i)[\sigma\rangle(PN, m_o)$ can be found such that $\langle \mu(\sigma_i)|\mu(\sigma_i) \neq s, i = 1 \dots |\sigma|\rangle = \sigma^L$. Transitions can also be mapped to a non-silent, blocking activity b (also not observed in the event log), which can hence never be executed during fitting replay of an event log trace.

3 Methodology

This section presents the developed real-time decomposed conformance analysis approach. Fig. 1 provides a schematic overview of the approach, which can be split up in four phases, explained in the next subsections. The implemented prototype has been implemented as a collection of ProM plugins[1].

3.1 Phase 1: Decomposition

The first phase of the proposed methodology entails *decomposition*. Formally, the overall system net $SN = (PN, m_i, m_o)$ is broken down into a collection of subnets $\{SN^1, SN^2, \dots SN^n\}$ such that the union of these subnets yields the original system net $SN = \bigcup_{1 \leq i \leq n} SN^i$. By means of decomposing the original model into a set of subnets we aim to achieve the following goals. First, fragment the conformance problems into a set of more comprehensive semantic elements aiding on the diagnosis. Second, restrict the possible pernicious effects of the heuristics decisions taken during the conformance analysis (see Phase 3 below). Third, speed-up the analysis compared with non-decomposed conformance checking techniques.

Due to the final goal of analyzing conformance, not all possible decomposition approaches are appropriate for this task. Only those *valid* decompositions that preserve the conformance integrity should be considered [1]. That is, given the original net and the decomposed version, *the original net perfectly conforms iff all the subnets in the decomposed setting perfectly conforms*. In other words, no conformance anomalies should be lost or introduced in the transition from the overall model to the decomposed one. In [1], the authors define a valid decomposition—applicable on Petri nets—as the decomposition that satisfies the following conditions:

1. Each arc of the overall net belongs to exactly one of the submodels, i.e.,$F = \bigcup_{1 \leq i \leq n} F^i$where $F^i \cap F^j = \emptyset$ for $1 \leq i < j \leq n$;
2. Each place of the overall net belongs to exactly one of the submodels, i.e.,$P = \bigcup_{1 \leq i \leq n} P^i$where $P^i \cap P^j = \emptyset$ for $1 \leq i < j \leq n$;

[1] ProM is an academic process mining framework to allow for rapid prototyping and plugin development. See: `http://www.processmining.org/prom/start`

Fig. 1. Architectural overview of the developed real-time decomposed conformance analysis technique.

3. Silent transitions appears in precisely one of the subnets, i.e., $\forall t \in T \setminus T_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$, where $T_v(SN)$ stands for the set of visible transitions (i.e., non-silent) of SN, i.e. $T_v(SN) = \{t \in T | \mu(t) \neq s\}$;
4. Non-silent, duplicate transitions appear in precisely one of the subnets, i.e., $\forall t \in T_v(SN) \setminus T_v^u(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$, where $T_v^u(SN)$ stands for the set of non-silent and non-duplicate transitions of SN, i.e. $T_v^u(SN) = \{t \in T | \mu(t) \neq s \wedge \nexists x \in T : x \neq t \wedge \mu(x) = \mu(t)\}$;
5. Non-silent, non-duplicate transitions may appear in multiple subnets, i.e., $\forall t \in T_v^u(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| \geq 1$.

In other words, all elements in the original Petri net model must belong to a submodel, but only non-silent, non-duplicate transitions can be shared among several submodels. In [1], the authors prove that any valid decomposition satisfying these conditions captures all the conformance problems of the overall model, and not more, i.e., it preserves the conformance integrity.

In [11], an approach based on SESE-components (Single-Entry Single-Exit) is presented, i.e. subgraphs in the workflow graph defined over a system net having single entry and exit boundary nodes [18]. The decomposition of the workflow graph of a Petri net into SESE-components is well-studied and provides a valid means to perform process model decomposition. In addition, SESE-components represent a well-defined and understandable part of the process model, with the added benefit that it is possible to define a hierarchical structure among the model fragments which allows to navigate through the different levels of granularity, so that a SESE-component perfectly reflects the idea of subprocesses within the main process. Fig. 2 depicts an example SESE-component for the illustrative case shown in Fig. 4, obtained using the technique proposed in [11]. Note that this techniques can be combined with a user-supervised post-processing step in order to obtain components that better fulfill the domain-aware monitoring.

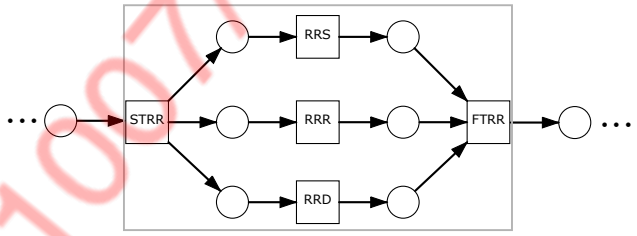

Fig. 2. *"Open and register transaction"* SESE-component from the case example in Fig. 5. *STRR* and *FTRR* are the *entry* and *exit boundary* nodes of the SESE-component, respectively. The rest of places and transitions are *interior* nodes of the SESE-component.

3.2 Phase 2: Event Dispatching

Once a system net has been decomposed into a set of submodels, this collection of models is passed to a central *event dispatcher*, which also serves to listen for incoming events. For each submodels, it is examined whether it contains a

transition t which maps to the incoming event e, i.e. $\exists t \in T : \mu(t) = act(e)$. If it does, this indicates that the event at hand should be replayed on this particular submodel (multiple such submodels can be found), and the event is passed forward to this model fragment.

It is possible to decouple the "worker"-instances for each model fragment in a distributed fashion, with each model fragment running on separate machines. For the purpose of our prototype, we have implemented a multi-threaded architecture where a number of worker threads smaller than or equal to the number of process fragments is spawned, with each worker thread overseeing the handling of one or more process fragments. This approach allows for the concurrent handling of event checking over the different model fragments.

3.3 Phase 3: Replay

Once it is determined which process model fragment(s) should parse the incoming event, the actual *replay* of this event on each such fragment is performed. For each process model fragment, a state list is maintained denoting the current marking reached by the currently-running process instances. When an event e is queued for replay by a process fragment, the state linked to process instance $id(e)$ is progressed by investigating whether there exists an enabled transition $\exists t \in T : \mu(t) = act(e) \wedge enabled(t)$. The outcome of this evaluation determines if the process model is showing discrepancies or not.

Some additional remarks should be provided at this point. First of all, we note that we apply a heuristic, event-granular replayer similar to the one applied in [19]. The reasoning behind the choice to opt for a replayer playing the token game instead of an alternative approach such as alignment or behavioral profile based techniques [3, 9, 8] are twofold. First, alignment and behavioral profile based replayers perform their analysis on a trace, rather than event level, meaning that a complete process instance needs to finalize in order to align the log trace with a process model transition sequence As we are dealing with event streams which need to be analyzed in a real-time manner, an event-granular replay strategy is required. Additionally, the behavioral profile approach does not specify how duplicate tasks can be dealt with. Second, alternative approaches suffer from scalability issues which make them unsuitable in a real-time context. Subsequently, the replay procedure applied here is formalized in Algorithm 1.

A second remark entails the way decision points are resolved by the replayer. Put briefly, whenever multiple (enabled) transitions are mapped to the same event log activity within a process model and/or whenever multiple invisible activities are enabled, the replayer needs to determine which transition to execute to handle the activity at hand. Note that—in extreme edge cases—it is possible that the forced firing of a non-enabled transition should be preferred if this avoids several other violations later in the event trace [20]. In Algorithm 1, a replay strategy is put forward which prefers the firing of enabled transition mapped to the activity at hand first, followed by the set of silent transitions, followed by the set of non-enabled transition mapped to the activity at hand. If the chosen set contains multiple transition candidates, a one-step look-ahead procedure is

executed to determine which candidate enables the execution of the following activity (if no such candidate can be found, a random one is chosen). For the multitude of process models, this look-ahead suffices to resolve any ambiguities. However, since we are dealing with streaming event data in this context, we possess no knowledge about events which will arrive in the future, preventing the execution of the look-ahead procedure. We propose and have implemented three methods to deal with this issue. First, disabling the look-ahead altogether and assuming that the model is deterministic enough to handle incoming events without taking the context into account ($n = 0$ in Algorithm 1). Second (another extreme), restarting the replay of the full trace each time an event is added, thus allowing the replayer to revise earlier decisions ($n = |\sigma^L|$ in Algorithm 1). Note however that the replayer is configured such that no new violations may be introduced related to historical activities ($\neg \mathit{conforms}(e) \vee \neg r$). In practice, this means that the replayer can revise the state chain by modifying the execution of silent transitions, selecting alternative albeit *also enabled* transition mapped to a particular activity for activities which were parsed correctly, or selecting alternative disabled transition, although only for activities which were not parsed correctly (provided by function *conforms* in Algorithm 1). The third method combines these two extremes by considering a part of the executed transition sequence as "frozen", only allowing revisions for the last n steps.

As a third remark, recall that it was mentioned in Subsection 2.1 that one of the drawbacks of "token game"-based replayers entails the possible creation of superfluous tokens, enabling subsequently for too much behavior. However, as was mentioned in the description of Phase 1, we note that the decomposition of a process model restricts the possible pernicious effects of the heuristics decisions taken during the conformance analysis, as each model is now limited to dealing with a smaller subset of behavior. In addition, as superfluous tokens are created following the forced firing of violating activities, the process instance or model fragment at hand is likely to be immediately indicated as "dubious" at this point, lowering the trustfulness of following events within this instance of model fragment, independent of the replay strategy being applied. In addition, recall that we are applying a hierarchical decomposition strategy, so that it is possible to perform the actual replay at a lower-granularity level than the visualization and reporting.

3.4 Phase 4: Reporting and Visualization

The final phase consists of reporting and visualization. Remark that, naturally, these actions can be performed while the actual conformance analysis is running. In general, two ways of result follow-up are supported by our architecture. The first one consists of the logging of various statistics by the running worker threads and replayers, which is polled regularly by decoupled components (e.g. a real-time dashboard or perhaps logged to a persistent data store). The second manner by which results can be interpreted consists of the definitions of various triggers which are to be fired once certain criteria are met, such as a model fragment overshooting a certain error rate threshold, for instance, of a high-risk activity or

Algorithm 1 Real-time event replay algorithm.

Input: $PN = (P, T, F)$, $SN = (PN, m_i, m_o)$ % Given Petri net and System net
Input: $e = (id, act, time)$ % Arriving event to be replayed (checked)
Input: σ^L % Trace of log activities having occurred so far for instance $id(e)$
Input: σ % Trace of model transitions being executed so far for instance $id(e)$
Input: m % Current marking of the model for instance $id(e)$
Input: $conforms : \sigma^L \to \{True, False\}$ % Function denoting conf. outcome of previous event
Input: $enabled : (T \times M) \to \{True, False\}$ % Function denoting if a transition is enabled under a given marking (M is set of all possible markings)
Input: $nextmarking : (T \times M) \to M$ % Function returning the marking after (force) firing a given transition in a given marking
Input: $random : (T' \subseteq T) \to T$ % Function returning random transition from a given set of transitions
Input: $\mu : T \to A \cup (s, b)$ % Mapping function between model and log
Input: $n := 0$ % Number of steps to revise in historic trace (default: 0, i.e. none)
Input: $r := False$ % Denoting whether the event being replayed is a revised historic event
Input: $e_{next} := \emptyset$ % Next incoming event (optional; used when revising earlier decisions)
Output: Executed transition $t_{replayed}$

function ReplayEvent(SN, e, e_{next}, σ^L, σ, m, n, r)
  % Handle revision of historic decision:
  if $\neg r \wedge n > 0$ **then**
    Remove last n items from σ and revert marking m to earlier state
    for $i \in \langle (n-1), \ldots, 0 \rangle$ **do** $ReplayEvent(SN, \sigma^L_{|\sigma^L|-i}, \sigma^L_{|\sigma^L|-i+1}, \sigma^L, \sigma, m, n, True)$
  end if
  % Set up transition candidate collections:
  $EC^m := \{t \in T | enabled(t, m) \wedge \mu(t) = act(e)\}$
  $EC^a := \{t \in T | enabled(t, m)\}$
  $EC^m_f := \{t \in EC^m | \exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e_{next})\}$
  $EC^a_f := \{t \in EC^a | \exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e_{next})\}$
  $I_f := \{t \in T | enabled(t, m) \wedge \mu(t) = s \wedge$
    $\exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e)\}$
  $t_{replayed} := \emptyset$ % Transition chosen to be fired
  % Determine transition to fire:
  if $|EC^m| > 0$ **then** % Transition is enabled
    if $|EC^m_f| > 0$ **then** $t_{replayed} := random(EC^m_f)$
    else $t_{replayed} := random(EC^m)$
  else if $|I_f| > 0$ **then** % Fire single invisible transition first
    $i' := random(I_f)$
    Update $\sigma := \langle \sigma, i' \rangle$
    Update $m := nextmarking(i', m)$
    $C := \{t \in T | enabled(t, m) \wedge \mu(t) = act(e)\}$
    $t_{replayed} := random(C)$
  else if $|EC| > 0 \wedge (\neg conforms(e) \vee \neg r)$ **then** % Force fire
    if $|EC_f| > 0$ **then** $t_{replayed} := random(EC_f)$
    else $t_{replayed} := random(EC)$
  else
    % No single transition is mapped to this event's activity, skip firing
  end if
  Update $\sigma := \langle \sigma, t_{replayed} \rangle$
  if $\neg r$ **then** Update $\sigma^L := \langle \sigma^L, e \rangle$
  Update $m := nextmarking(t_{replayed}, m)$
  return $t_{replayed}$
end function

function ReplayTrace(SN, τ)
  % Function added for completeness, not used in real-time setting
  $m := m_i$
  $\sigma^L := \langle \rangle$
  $\sigma := \langle \rangle$
  for $i \in 1..|\tau|$ **do** $ReplayEventSN$, τ_i, τ_{i+1}, σ^L, σ, m, 0, $False$
  return σ
end function

model fragment being violated. The actions which can be undertaken as a result are self-explanatory, e.g. sending warnings, or halting running process instances or even the complete system.

3.5 Implementation

Fig. 3 depicts a screen capture of our developed proof-of-concept implementation is shown. In the prototype, events are streamed over a network in real-time using a separate program (shown as the top left window in Fig. 3), which are received by the conformance analyzer and verified against the decomposed model fragments. The top panel displays a global overview of the model being checked against, with violating parts highlighted. Since the analysis is performed on the basis of the decomposed model fragments, it is more straightforward to pinpoint errors to a localized area within the global view than when using the full model as-is to perform conformance analysis. The lower left panels depict error monitors per submodel, showing the error rate for each model fragment over time. The panel on the right shows general statistics and program information. Note that this real-time approach allows to immediately react once a certain (user-configurable) criteria are triggered, such as model fragments (or specific activities) reaching a certain failure threshold., and shows the error rate for each model fragment together with a global overview for the complete process model.

4 Case Example

In this section we propose the study of a realistic process case example in order to illustrate the approach presented in this paper and its benefits. Model and logs—original and decomposed—of this case example, together with the rest of the benchmarks used in this experimental section, are publicly available[2].

4.1 Description

The modeled process describes a realistic transaction process within a banking context. The process contains all sort of monetary checks, authority notifications, and logging mechanisms responding to the new degree of responsibility and accountability that current economic environments demand. The process is structured as follows (Fig. 4 shows a high-level overview of the complete process): it is initiated when a new transaction is requested, opening a new instance in the system and registering all the components involved. The second step is to run a check on the person (or entity) origin of the monetary transaction. Then, the actual payment is processed differently, depending of the payment modality chosen by the sender (cash, cheque[3] and payment). Later, the receiver is checked and the money is transferred. Finally, the process ends registering the information, notifying it to the required actors and authorities, and emitting the corresponding receipt.

[2] doi:10.4121/uuid:c1d1fdbb-72df-470d-9315-d6f97e1d7c7c

[3] The British term is used to avoid ambiguity with the verb "to check".

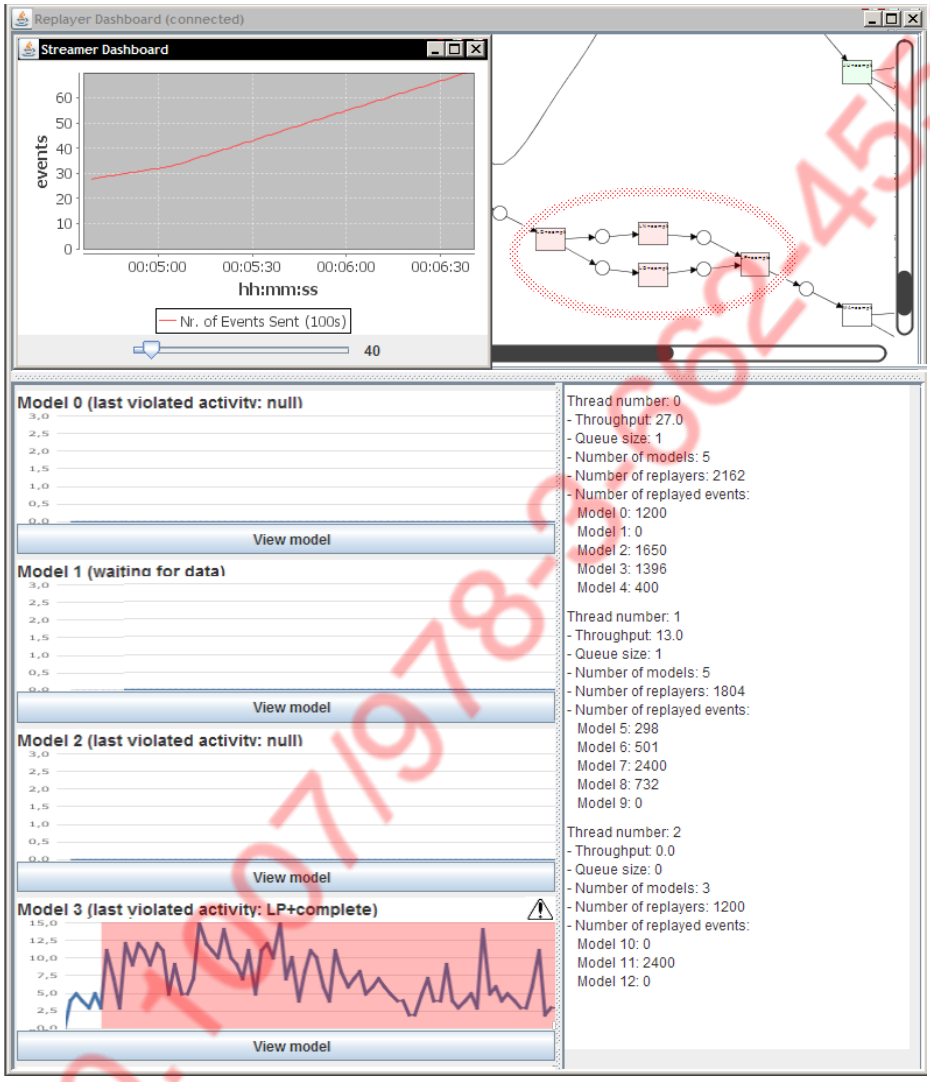

Fig. 3. Screen capture of the developed real-time event conformance analysis prototype. A global overview of the model being checked against, error rates per submodel, and general statistics are reported. Our real-time approach allows to immediately react once a certain (user-configurable) criteria are triggered, such as model fragments (or specific activities) reaching a certain failure threshold.

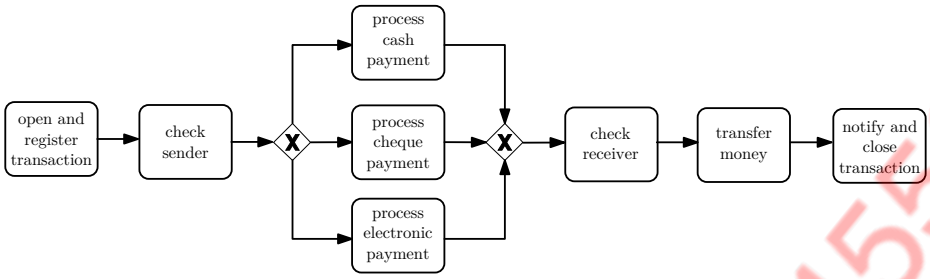

Fig. 4. High level overview of the running example process, structured in subprocesses.

The process has been modeled in terms of a Petri net. The decomposition techniques based on SESE-components (see Section 3) is used to decompose the overall model into suprocesses. In particular, a valid decomposition where components have size at most 60 is derived. Finally, the decomposition is post-processed by merging some of the SESE-components in order to reach the final decomposition shown in Fig. 5 (which depicts the full process): eight of the proposed subnets correspond with the eight subprocesses identified in Fig. 4 (represented within gray rectangles), and the ninth subnet contains all the trivial connections between suprocesses (represented outside the rectangles).

4.2 Experimental Scenario Evaluation

To illustrate the benefits of the technique, we present two possible scenarios within the case example process.

Scenario 1: Serial Number Check The modeled process defines that, whenever a client executes the payment in cash, the serial numbers must be checked (see Fig. 4). The banking regulation states that serial numbers must be compared with an external database governed by a recognized international authority (*"Check Authority Serial Numbers CASN"*). In addition, the bank of the case example decided to incorporate two complementary checks to its policy: an internal bank check (*"Check Bank Serial Numbers CBSN"*), and a check among the databases of the bank consortium this bank belongs to (*"Check Inter-Bank Serial Numbers CIBSN"*). At a given point, due to technical reasons (e.g., peak hour network congestion, malfunction of the software, deliberated blocking attack, etc.), the external check *CASN* is not longer performed, contradicting the modeled process, i.e., all the running instances of the process involving cash payment can proceed without the required check. Using the proposed approach, this situation is detected immediately, identifying the anomalous subprocess (*process cash payment*), focusing the conformance analysis on it, and eventually taking the necessary countermeasures. The consequences of detecting such cases only in forensic analysis performed months after the incident are severe and difficult to recover from. The situation is depicted in Fig. 6.

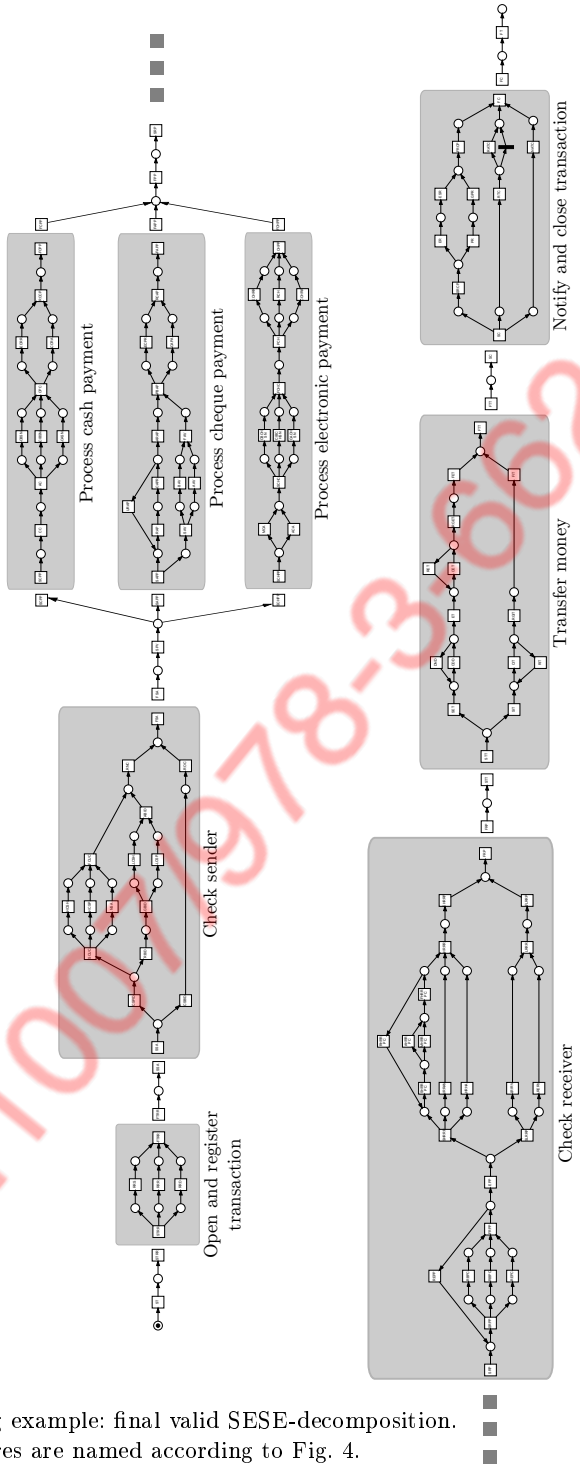

Fig. 5. Running example: final valid SESE-decomposition. The substructures are named according to Fig. 4.

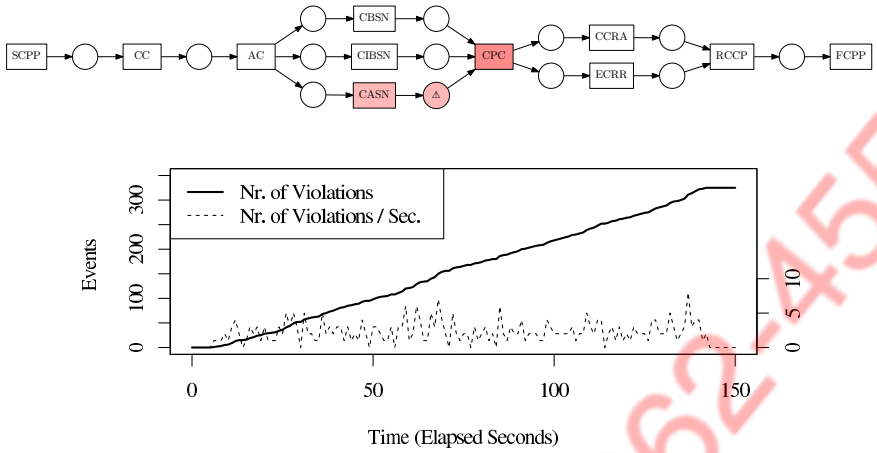

Fig. 6. In the first scenario, the *Check Authority Serial Number* (*CASN*) activity is skipped for some process instances, causing the *CPC* activity to fail, due to a missing input token which was expected to be present and placed there by the execution of *CASN*. The figure depicts the error localized in the affected model fragment; the graph depicts the cumulative and running amount of violations detected within this fragment.

Scenario 2: Receiver Preliminary Profiling During the *check receiver* stage, the model establishes two steps to be performed sequentially: first, a preliminary profiling analysis (*"Start Receiver Pre Profiling SRPP"*) is executed over the receiver in order to evaluate and establish its potential risk (*"Evaluate Pre Profiling EPP"*). Only then, a complete background check is performed over the receiver, where this check can either be more casual (*"Start Low Risk Receiver Processing SLRRP "*) or thoroughly (*"Start High Risk Receiver Processing SHRRP"*) depending on the potential risk detected on the preliminary profiling. However, the presence of an inexperienced bank employee, malevolence, or simply a bad implemented bank evaluation protocol, could result in evaluating the receiver with an unfinished preliminary profile check. The situation is depicted in Fig. 7.

4.3 Experimental Comparison

To benchmark the performance of our developed real-time conformance analysis technique against related approaches, a fitting event log was generated (based on the model depicted in Fig. 5) containing ten thousand process instances (678864 events). A non-conforming ("noisy") variant of this event log was produced by inducing noise (inserting, deleting, and swapping of events) so that 10% of the included events are erroneous.

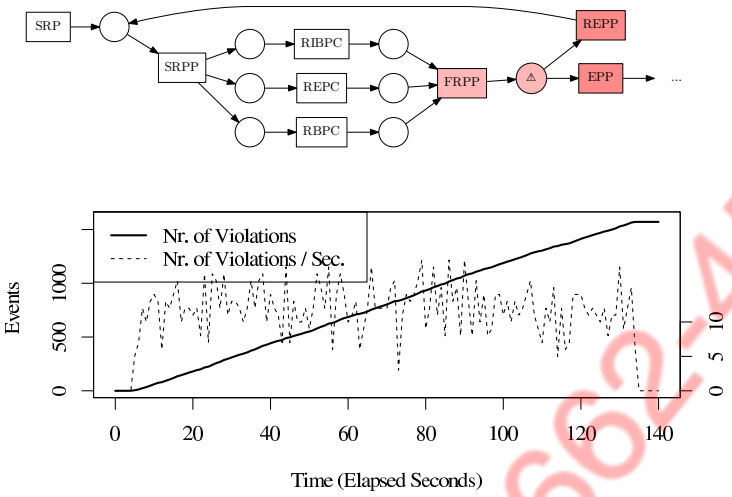

Fig. 7. In the second scenario, the preliminary profile check for receivers is skipped (*SRPP* to *FRPP*), causing either the *REPP* or *EPP* activities to fail. The figure depicts the error localized in the affected model fragment; the graph depicts the cumulative and running amount of violations detected within this fragment.

We compare our proposed technique against the alignment based replay technique by Adriansyah et al. [3] as well our original implementation of the token-game based heuristic replayer [19]. Both the non-decomposed and decomposed variants of these techniques were included, applying hereto the methodology as described in [12].

Fig. 8 depicts the performance results of the experiment, showing the amount of time taken (x-axis) to check the conformance of the included event logs (the y-axis represents the cumulative ratio of event checks performed). As can be seen, our proposed real-time conformance analysis technique performs competitively with respect to related techniques. During the experimental run, a maximum throughput rate (number of events checked per second) was reached at 35000 with the experiment running on a single consumer laptop with three worker threads. Some additional remarks should be provided however when interpreting Fig. 8. First, note that our proposed technique performs similarly as the heuristic decomposed replay technique, but note that our techniques executes a *conformance check on an event-granular basis* and thus can be applied in a real-time monitoring setting, whereas the other techniques do so on a trace-granular level (i.e. a complete trace should be provided to perform the replay procedure). However, the event log is of sufficient size so that a step-wise effect is not apparent in Fig. 8. Second, the replay procedure of the existing techniques was modified such that *each trace is checked independently* of the log context, meaning that no distinct trace grouping is performed over the log and each trace is checked as if it were belonging to an event log containing only this trace, so as to

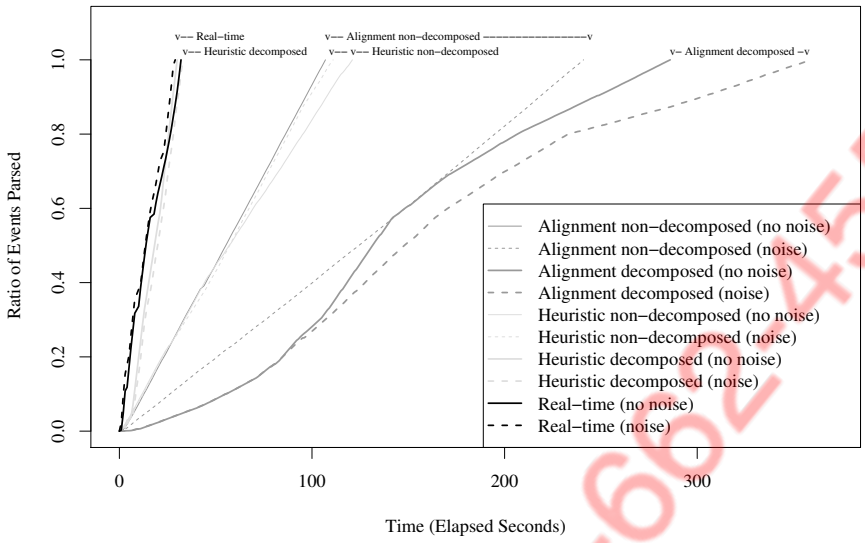

Fig. 8. Comparison of replay performance for the included techniques in the experimental setup, showing the time taken per technique to replay the given event log.

better assess the performance of these techniques in a real-time scenario (where the complete trace and log are unknown as events are arriving), rather than a post-hoc scenario where the complete event log is provided as-is. Note that—for the alignment based technique—this causes the non-decomposed version to perform better than the decomposed one. This is a perhaps unexpected result, but is caused by the fact that the alignment based techniques are geared towards checking—and as such expect—event logs as a whole. We thus emphasize the fact that these techniques have—currently—not been optimized to be applied in a real-time scenario (with an event stream being checked instead of an historical log).

5 Conclusions and Future Work

In this paper, we have presented a novel business process conformance analysis technique which is able to support real-time monitoring of event-based data streams. Our approach offers a number of novel contributions, most notably a speed-up compared to related techniques, the ability to localize discrepancies and allowing real-time monitoring and thus rapid response times in mission-critical or high-risk environments, which is a significant benefit compared to existing conformance checking techniques which mainly work in an offline manner.

Future lines of research include: streamlining visualization and reporting capabilities of our prototype, incorporating other decomposition and replay strate-

gies, and adapting the framework into a distributed implementation, where different replayer engines run on separate machines. In addition, we plan to pursue additional real-life case studies to confirm the validity of our approach.

Acknowledgments. This work is supported by the KU Leuven research council (grant OT/10/010), the Flemish Research Council (Odysseus grant B.0915.09), FPU grant (AP2009-4959) and project FORMALISM (TIN-2007-66523).

References

[1] van der Aalst, W.M.P.: Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507
[2] Adriansyah, A., Muñoz-Gama, J., Carmona, J.: Alignment Based Precision Checking. BPM Center Report, 12-10-2012 (2012)
[3] Adriansyah, A., van Dongen, B., van der Aalst, W.: Towards Robust Conformance Checking. In: BPM 2010 International Workshops and Education Track (BPM 2010), volume 66 of Lecture Notes in Business Information Processing. (2011) 122–133
[4] Adriansyah, A., van Dongen, B.F.: Conformance checking using cost-based fitness analysis. In: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2011). (2011) 55–64
[5] Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In Rosa, M.L., Soffer, P., eds.: Business Process Management Workshops. Volume 132 of Lecture Notes in Business Information Processing., Springer (2012) 137–149
[6] van der Aalst, W., Adriansyah, A.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining and Knowledge Discovery **2** (2012) 1–18
[7] Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1) (2008) 64–95
[8] Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. **37**(3) (2011) 410–429
[9] Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In Maglio, P.P., Weske, M., Yang, J., Fantinato, M., eds.: ICSOC. Volume 6470 of Lecture Notes in Computer Science. (2010) 1–16
[10] van der Aalst, W.M.P.: Decomposing process mining problems using passages. In Haddad, S., Pomello, L., eds.: Petri Nets. Volume 7347 of Lecture Notes in Computer Science., Springer (2012) 72–91
[11] Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Conformance checking in the large: Partitioning and topology. In Daniel, F., Wang, J., Weber, B., eds.: BPM. Volume 8094 of Lecture Notes in Computer Science., Springer (2013) 130–145
[12] Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Hierarchical conformance checking of process models based on event logs. In Colom, J.M., Desel, J., eds.: Petri Nets. Volume 7927 of Lecture Notes in Computer Science., Springer (2013) 291–310
[13] Gal, A., Hadar, E.: Generic architecture of complex event processing systems. In Hinze, A., Buchmann, A.P., eds.: Principles and Applications of Distributed Event-Based Systems. IGI Global (2010) 1–18
[14] Zarri, G.P.: Representation and processing of complex events. In: AAAI Spring Symposium: Intelligent Event Processing, AAAI (2009) 101–
[15] Kang, B., Lee, S.K., bin Min, Y., Kang, S.H., Cho, N.W.: Real-time process quality control for business activity monitoring. In Gavrilova, M.L., Gervasi, O., Taniar, D., Mun, Y., Iglesias, A., eds.: ICCSA Workshops, IEEE Computer Society (2009) 237–242
[16] Janiesch, C., Matzner, M., Müller, O.: Beyond process monitoring: a proof-of-concept of event-driven business activity management. Business Proc. Manag. Journal **18**(4) (2012) 625–643
[17] Murata, T.: Petri nets: Properties, analysis and applications. In: Proceedings of the IEEE. Volume 77-4. (1989) 541–580
[18] Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In Bravetti, M., Bultan, T., eds.: WS-FM. Volume 6551 of Lecture Notes in Computer Science., Springer (2010) 25–41
[19] vanden Broucke, S.K., Weerdt, J.D., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Transactions on Knowledge and Data Engineering **99**(PrePrints) (2013) 1
[20] vanden Broucke, S., De Weerdt, J., Vanthienen, J., Baesens, B.: On replaying process execution traces containing positive and negative events. Feb research report kbi 1311, KU Leuven (2013)