

RESEARCH ARTICLE

Diagnosing Conformance Between Object-Centric Event Logs and Models

BAOXIN XIU¹ AND GUANGMING LI² ¹School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China²PLA No. 31307, Chengdu 610051, China


Corresponding author: Guangming Li (kfgz087@126.com)

ABSTRACT Conformance checking techniques can diagnose discrepancies between the behavior observed in an event log and the behavior allowed by a reference model. These discrepancies might indicate undesirable deviations, fraud, inefficiencies, or other issues in a business process. Today, most organizations use object-centric systems such as ERP and CRM systems, which generate and store data in an object-centric manner. Unfortunately, existing conformance checking techniques often employ process-centric models, e.g., Petri nets, which typically consider process instances in isolation (ignoring interactions among them) and are more focused on the behavioral perspective of processes. Accordingly, the conformance diagnosis results are only on the behavioral perspective and cannot deal deviations related to the data perspective and interactions, failing to reveal some useful insights related to the undiscovered deviations. In this paper, we aim to check conformance based on object-centric logs and models, which combines data and behavior perspectives, and the resulting diagnostic information contains conformance problems that would have remained undetected using conventional techniques. At last, our conformance checking approach is evaluated based on the data generated in a real ERP system.

INDEX TERMS Process mining, conformance checking, object-centric, OCEL, OCBC models, ERP.

I. INTRODUCTION

Conformance checking compares event logs to process models to detect discrepancies between the modeled behavior and the observed behavior. Companies often use process models (explicit or implicit) to describe how their business processes should be executed. However, in flexible business environments, these models usually give suggestions to indicate how to operate information systems and it is possible to violate the predefined processes [1]. Therefore, there may exist conformance problems between predefined processes and real executions [2]. For companies, it is necessary to know where the real behavior violates the predefined process and get rid of the violations when they correspond to unexpected situations. Conformance checking is relevant for business alignment and auditing. For example, the event log can be replayed on top of the process model to find undesirable deviations suggesting fraud or inefficiencies.

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar .

Event logs and process models are the starting point to apply conformance checking techniques. Classic logs and models assume a single case notion for the business process. However, in reality, most organizations use object-centric systems such as ERP and CRM systems, which generate and store data in an object-centric manner. In these processes various objects interact and one event may correspond to multiple types of objects, e.g., orders, invoices, and packages. With selecting one of the object types as a case notion, classic logs and models are flattened and can only provide a specific view on the process, leading to convergence and divergence problems. Besides, they typically consider process instances in isolation (ignoring interactions among them) and are more focused on the behavioral perspective of processes [3], [4], [5].

The existing conformance checking techniques often employ flattened logs and models [1], [6], [7], [8], [9]. Accordingly, when applying existing techniques on the object-centric systems, the diagnosis results are only on the behavioral perspective and fail to deal deviations related

to the data perspective and interactions. Therefore, some useful insights (related to the undiscovered deviations) cannot be revealed. Fortunately, a few logging formats have been proposed to tackle the problem. [10] extracts eXtensible Object-Centric (XOC) event logs to check conformance. These techniques are not widely adopted because of their complexity and performance problems (e.g., XOC allows reconstructing the entire database state).

Since the deviations related to the data perspective and interactions are important, in this paper we check conformance based on an object-centric process model (named the *Object-Centric Behavioral Constraint (OCBC)* model [11], [12]) and an object-centric event log (OCEL) [13]. Unlike existing approaches, in an object-centric manner, instances are not considered in isolation and the data perspective is taken into account. Hence, we can now detect and diagnose a range of conformance problems that would have remained undetected using conventional logs and models.

This paper is organized as follows. Section II discusses the related work. In Section III, we define OCEs and OCBC models, which are taken as the input of conformance checking techniques. Section IV checks conformance on the data perspective, behavioral perspective and interactions in between, respectively. In Section V, we evaluate our conformance checking approach and compare it with other conformance checking techniques. Section VI concludes this paper.

II. RELATED WORK

Traditional conformance checking techniques (and similar techniques such as compliance checking, auditing, Six Sigma, etc.) are proposed to diagnose the conformance problems, i.e., if reality, as recorded in the log, conforms to the model and vice versa.

The first type of conformance checking techniques are based on replaying the traces in the event log on the model, i.e., token-based replay techniques [1], [6], [7]. They use both an event log and a process model as input, i.e., history is replayed on the model to analyze various phenomena. More precisely, the numbers of produced, consumed, missing and remaining tokens are counted while replaying the event log, which is used to diagnose conformance problems on the control-flow perspective. For instance, if an activity in the event log is not enabled, then a missing token is added. These numbers can be used to diagnose conformance problems. Token-based replay can differentiate between fitting and non-fitting cases. It is easy to understand and can be implemented efficiently. However, this approach has some drawbacks. Intuitively, fitness values tend to be too high for extremely problematic event logs. If there are many deviations, the Petri net gets “flooded with tokens” and subsequently allows for any behavior. The approach is also Petri-net specific and can only be applied to other representations after conversion.

Alignment-based approaches are state-of-the-art techniques, which are introduced to overcome these limitations [8], [9], [14], [15]. Given a trace in the event log, the closest path in the model is computed by solving an optimization problem. In alignment approaches, model moves and log moves are used to describe the misalignment, and they are easier to interpret than missing and remaining tokens. By assigning a cost function to this misalignment, fitness is computed based on the total costs. Based on these techniques, it is possible to quantify the level of conformance. Most techniques focus on fitness, however, there are also techniques for computing other quality dimensions such as simplicity, precision, and generalization [16].

In [16] a prefix automaton is constructed based on the event log to count so-called escaping edges. By quantifying these edges and their frequency, it provides an accurate measurement of the precision dimension. [17] extends and complements the technique introduced in [16], by additionally considering potential variability in the event log. Moreover, it introduces a novel technique to estimate the confidence interval of the metric, which estimates the robustness of the precision. Note that traces in the log do not need to be completely fitting. In [18], by aligning the event log and the model, the pre-alignment makes it possible to measure precision more accurately, even in case of deviations. [19] shows how to compute the generalization criterion, which measures if the model is “overfitting”.

In addition to checking conformance on Petri nets, [20], [21] check the conformance of artifact-centric models expressed in terms of proclets. Also related is the work on conformance checking of *declarative models* [22], [23]. [22] proposes an approach to compute the fitness of Declare models. It creates a constraint automata for each constraint to derive the optimal alignment. It also presents some techniques to remove the search space. [23] discusses how to compute fitness, precision and generalization for Declare models. However, this work does not consider multiple intertwined instance notions and also does not relate control-flow to some overall data model.

There also exist approaches which check conformance more than the control-flow perspective. [24] constructs a timed business process model describing activities from mobile services, and checks conformance by verifying how compliant the constructed model and a new arrival event log. [25] extends BPMN to support modeling resources and their pricing strategies. Then the matching between temporal constraints of Cloud resources and the temporal constraints of BP activities are checked. These two approaches check the conformance on the control-flow and time perspectives in terms of temporal constraints. [26] describes an approach that aligns event log and model, and takes data and resources into account when checking process conformance. Multi-perspective conformance checking [27], [28] considers the conformance of process to multiple perspectives of a process model (i.e., control-flow, data, resources, time) at the

same time. [10] can check conformance on behavioral and data perspectives based on XOC logs. However, the XOC logs reconstruct the entire database state, which are too complex and suffer performance problems.

III. PRELIMINARY

A. PETRI NETS

Petri nets are the best-investigated process modeling language allowing for the modeling of concurrency. Petri nets use a very simple notation of circles representing places and squares representing transitions with arrows connecting them. Although the graphical notation is intuitive and simple, Petri nets are executable and many analysis techniques can be used to analyze them [29].

More precisely, a Petri net is a bipartite graph consisting of places and transitions. The network structure is static, but tokens (presented by black dots) can flow through the network, governed by the firing rule illustrated next. A transition can represent a task and when executed it consumes one token from each of its input places and produces a token in each of its output places. In this way, tokens are moved between places and the state of a Petri net is determined by the distribution of tokens over places. The distribution is referred to as the marking, and the initial (final) marking indicates the start (end) of the process.

A Petri net $N = (P, T, F)$ defines a directed graph with nodes $N = (P \cup T)$ and edges F . The firing rule defines the dynamic behavior of a marked Petri net. A transition $t \in T$ is enabled in a marking M of net N , denoted as $(N, M)[t]$ if each of its input places t contains at least one token. An enabled transition t may fire, i.e., one token is removed from each of the input places t and one token is produced for each of the output places t . $(N, M)[t](N, M')$ denotes that t is enabled in M and firing t results in the marking M' .

B. OBJECT-CENTRIC EVENT LOGS

In general, event logs are used to record past events related to (object-centric) information systems, such as EPR and CRM systems. An event log is defined to record operations and the entities modified by operations related to a business process, such as Order-to-Cash (O2C) and Purchase-to-Pay (P2P) processes in ERP systems.

More precisely, operations executed on the systems are represented by events and physical and informational entities composing business processes are represented by objects, such as materials, documents, products, invoices, etc. An event is associated with an identifier, an activity, and a timestamp [30]. The activity and timestamp represent what happens in the execution and when it happens, respectively. Events may have attributes such as the resource executing the corresponding event. Events are atomic and ordered. For simplicity, we assume a total order. Note that the execution of an event involves a set of related objects. Each object has an identifier and is associated with an object type (i.e., object class). Objects may have attributes such as the customer of

an order. Objects often have related objects, e.g., an order has several order items.

Definition 1 (Universes): Below are the universes used in the formal definition in this paper:

- \mathcal{U}_O is the universe of object identifiers.
- \mathcal{U}_C is the universe of object classes or object types.
- $\mathcal{U}_R = \mathcal{U}_C \times \mathcal{U}_C$ is the universe of relationships between classes.
- \mathcal{U}_E is the universe of events.
- \mathcal{U}_A is the universe of activities.
- \mathcal{U}_{timest} is the universe of timestamps.
- \mathcal{U}_{Attr} is the universe of attribute names.
- \mathcal{U}_{Val} is the universe of attribute values.
- \mathcal{U}_{Card} is the universe of cardinalities.
- \mathcal{U}_{Con} is the universe of constraints.
- $\mathcal{U}_{CT} = \{X \subseteq \mathbb{N} \times \mathbb{N} \mid X \neq \emptyset\}$ is the universe of constraint types.

Definition 2 (Object Centric Event Log): An *object centric event log (OCEL)* is a tuple $L = (E, O, \pi_{act}, \pi_{time}, \pi_{evmap}, \pi_{eomap}, \pi_{otyp}, \pi_{ovmap}, \pi_{oomap} \preceq)$, where

- $E \subseteq \mathcal{U}_E$ is a set of event identifiers,
- $O \subseteq \mathcal{U}_O$ is a set of object identifiers,
- $\pi_{act} \in E \rightarrow \mathcal{U}_A$ maps events onto activities,
- $\pi_{time} \in E \rightarrow \mathcal{U}_{timest}$ maps events onto timestamps,
- $\pi_{evmap} \in E \rightarrow (\mathcal{U}_{Attr} \nrightarrow \mathcal{U}_{Val})$ maps events onto a partial function assigning values to some attributes,¹
- $\pi_{eomap} \in E \rightarrow \mathcal{P}(O)$ associates events to objects,
- $\pi_{otyp} \in O \rightarrow \mathcal{U}_C$ maps objects onto object types (or classes),
- $\pi_{ovmap} \in O \rightarrow (\mathcal{U}_{Attr} \nrightarrow \mathcal{U}_{Val})$ maps objects onto a partial function assigning values to some attributes,
- $\pi_{oomap} \in O \rightarrow \mathcal{P}(O)$ associates an object to a set of related objects, and
- $\preceq \subseteq E \times E$ defines a total order on events.²

\mathcal{U}_L is the universe of OCELS.

An OCEL is a collection of events that belong together, i.e., they belong to some “process” where many types of objects/instances may interact. Note that one event may refer to one or multiple objects and one object may be referred to by one or multiple events. The objects referred to by an event indicate that they are impacted by the operation corresponding to the event. Such an event log is object-centric since the events are related through the data perspective. Table 1 and Table 2 briefly present an object-centric event log [31], [32]. Table 1 represents the event records, where each row corresponds to a distinct event. Table 2 represents the relevant information of objects in the information systems.

C. OBJECT-CENTRIC CONSTRAINT BEHAVIORAL MODEL

An OCBC model combines data/object modeling techniques (ER, UML, or ORM) and a declarative process modeling

¹ $f \in X \nrightarrow Y$ is a partial function with domain $dom(f) \subseteq X$.

²A total order is a binary relation that is (1) antisymmetric, i.e. $e_1 \leq e_2$ and $e_2 \leq e_1$ implies $e_1 = e_2$, (2) transitive, i.e. $e_1 \leq e_2$ and $e_2 \leq e_3$ implies $e_1 \leq e_3$, and (3) total, i.e., $e_1 \leq e_2$ or $e_2 \leq e_1$.

language (inspired by *Declare* [33]). More precisely, an OCBC model consists of a class model (presenting cardinality constraints between objects), a behavioral model (presenting declarative constraints between events) and so-called AOC relationships which connect these two models by relating activities in the behavioral model to object classes in the class model.

Definition 3 (Class Model): A class model is a tuple $ClM = (C, R, \sharp_{src}, \sharp_{tar})$, where

- $C \in \mathcal{U}_C$ is a set of classes,
- $R \in \mathcal{U}_R$ is a set of relationships,
- $\sharp_{src} \in R \rightarrow \mathcal{U}_{Card}$ gives the source cardinality of a relationship (i.e., $\sharp_{src}(r)$ gives the cardinality on the c_1 side for $r = (c_1, c_2) \in R$), and
- $\sharp_{tar} \in R \rightarrow \mathcal{U}_{Card}$ gives the target cardinality of a relationship (i.e., $\sharp_{tar}(r)$ gives the cardinality on the c_2 side for $r = (c_1, c_2) \in R$).

\mathcal{U}_{ClM} is the universe of class models.

A behavioral temporal restriction may cover two perspectives, i.e., it may restrict the number of target events both before and after the reference event. In this paper, we will employ a graphical notation, i.e., a set of constraint types inspired by *Declare* (a declarative workflow language [33]), to represent and visualize the restrictions.

Any element of \mathcal{U}_{CT} is a constraint type which specifies a non-empty set of pairs of integers (*before*, *after*): the first integer *before* defines the number of target events before the reference event and the second integer *after* defines the number of target events after the reference event. Figure 1 shows the graphical notations of 8 example constraint types.

- The constraint type “response” means that there should be at least one target event after the reference event, i.e., $before \geq 0$ and $after \geq 1$.
- The constraint type “precedence” means that there should be at least one target event before the reference event, i.e., $before \geq 1$ and $after \geq 0$.
- The constraint type “co-existence” means that there should be at least one target event before or after the reference event, i.e., $before + after \geq 1$.
- The constraint type “unary-response” means that there should be precisely one target event after the reference event, i.e., $before \geq 0$ and $after = 1$.
- The constraint type “unary-precedence” means that there should be precisely one target event before the reference event, i.e., $before = 1$ and $after \geq 0$.
- The constraint type “non-existence” means that there should be zero target event before and after the reference event, i.e., $before = 0$ and $after = 0$.
- The constraint type “non-response” means that there should be zero target event after the reference event, i.e., $before \geq 0$ and $after = 0$.
- The constraint type “non-precedence” means that there should be zero target event before the reference event, i.e., $before = 0$ and $after \geq 0$.

Definition 4 (Activity Model): An activity model is a tuple $ActM = (A, Con, \pi_{ref}, \pi_{tar}, type)$, where

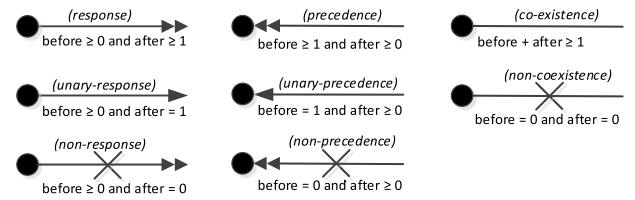


FIGURE 1. Graphical notation for 8 example constraint types. The dot on the left-hand side of each constraint refers to the reference events. Target events are on the other side that has no dot. The notation is inspired by *Declare*, but formalized in terms of cardinality constraints rather than LTL.

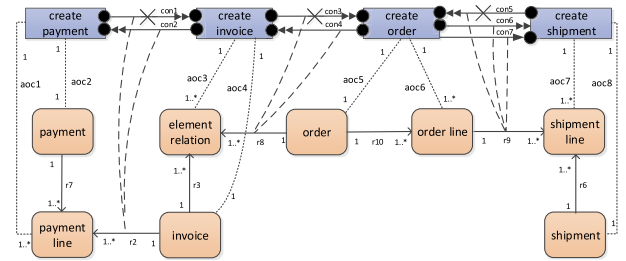


FIGURE 2. An OCBC model describing the order-to-cash process in ERP systems.

- $A \subseteq \mathcal{U}_A$ is a set of activities (denoted by rectangles),
- $Con \subseteq \mathcal{U}_{Con}$ is a set of constraints ($A \cap Con = \emptyset$, denoted by various types of edges),
- $\pi_{ref} \in Con \rightarrow A$ defines the reference activity of a constraint (denoted by a black dot connecting constraint and activity),
- $\pi_{tar} \in Con \rightarrow A$ defines the target activity of a constraint (other side of edge), and
- $type \in Con \rightarrow \mathcal{U}_{CT}$ specifies the type of each constraint (denoted by the type of edge).

\mathcal{U}_{ActM} is the universe of activity models.

Definition 5 (AOC Relationships): Let $A \in \mathcal{U}_A$ be a set of activities, $C \in \mathcal{U}_C$ be a set of classes. $AOC \subseteq A \times C$ is a set of AOC relationships between Activities and (Object) Classes. For convenience, we define three functions to refer to the cardinalities on the relationships.

- $\sharp_A \in AOC \rightarrow \mathcal{U}_{Card}$ gives the source cardinality of an AOC relationship (activity side), and
- $\sharp_{OC} \in AOC \rightarrow \mathcal{U}_{Card}$ gives the target cardinality of an AOC relationship (object class side).

\mathcal{U}_{AOC} is the universe of AOC relationships.

Definition 6 (Object-Centric Behavioral Constraint Model): An object-centric behavioral constraint model is a tuple $OCBCM = (ClM, ActM, AOC, \sharp_A, \sharp_{OC}, crel)$, where

- $ClM = (C, R, \sharp_{src}, \sharp_{tar})$ is a class model (Definition 3),
- $ActM = (A, Con, \pi_{ref}, \pi_{tar}, type)$ is an activity model (Definition 4),
- C, R, A and Con are pairwise disjoint (no name clashes),
- $AOC \subseteq A \times C$ is a set of AOC relationships, and \sharp_A and \sharp_{OC} specify the cardinalities on the AOC relationships (Definition 5),

- $crel \in Con \rightarrow C \cup R$ indicates the event correlation pattern (to identify the scope) for each behavioral constraint, satisfying the following conditions for each $con \in Con$:
 - $\{(\pi_{ref}(con), c), (\pi_{tar}(con), c)\} \subseteq AOC$ if $crel(con) = c \in C$, and
 - $\{(\pi_{ref}(con), c_1), (\pi_{tar}(con), c_2)\} \subseteq AOC$ or $\{(\pi_{ref}(con), c_2), (\pi_{tar}(con), c_1)\} \subseteq AOC$ if $crel(con) = (c_1, c_2) \in R$.

\mathcal{U}_{OCBCM} is the universe of OCBC models.

Figure 2 shows an OCBC model which describes the *order-to-cash* scenario in ERP systems. The model indicates that there are eight classes and four activities involved in this process. The class relationships reveal the constraints between classes, e.g., each order line should have a corresponding shipment line indicated by $r9$ (this is consistent with the real scenario where each order line is shipped to the corresponding customer). The seven behavioral constraints (i.e., $con1 \sim con7$) present restrictions assigned on the temporal order between events of different activities. For instance, $con6$ indicates that each “create order” event is followed by one or more corresponding “create shipment” events while $con7$ requires each “create shipment” event is preceded by precisely one corresponding “create order” event. The eight AOC relations (i.e., $aoc1 \sim aoc8$) specify the cardinality constraints between activities and classes. For example, $aoc5$ shows a one-to-one correspondence between “create order” events and “order” objects, i.e., if an “order” object is observed, the corresponding “create order” activity needs to be executed once and vice versa.

IV. DIAGNOSING CONFORMANCE

In this section, we diagnosing conformance based on an OCEL L and an OCBC model M . More precisely, we check whether reality (in the form of events and objects in L) conforms to the model M from three perspectives, i.e., the data perspective, the behavioral perspective and the interactions between them. In the diagnosis, seven types of conformance problems are identified. Most of these problems are not captured by existing conformance checking approaches [1], [19], [21], [26].

A. CONFORMANCE CHECKING ON DATA PERSPECTIVE

In the object-centric logs and models, the data perspective serves as the backbone. Accordingly, we first introduce how to check the conformance on the data perspective, i.e., check if each object conforms to the class model.

Definition 7 (Conformance on Data Perspective): Let L be an OCEL and M be an OCBC model. L conforms to M on the data perspective if and only if the following rules are satisfied:

- **Each object refers to objects of right classes (Rule I):**
for any $o_1 \in O$ and $o_2 \in \pi_{oomap}(o_1)$, we have that

$$(\pi_{otyp}(o_1), \pi_{otyp}(o_2)) \in R \quad (1)$$

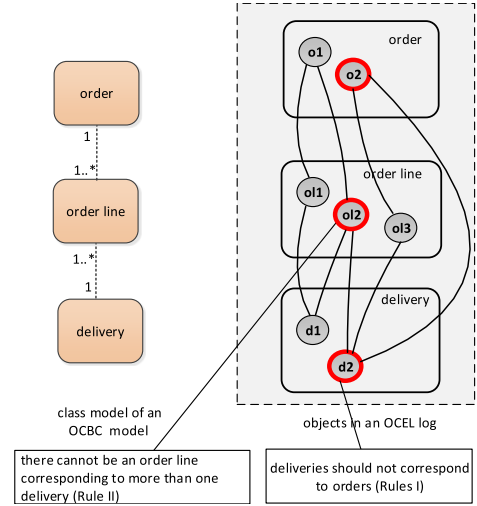


FIGURE 3. An illustration for checking conformance on the data perspective (detecting violations of data cardinality constraints).

or

$$(\pi_{otyp}(o_2), \pi_{otyp}(o_1)) \in R, \quad (2)$$

and

- **Each object has the right number of related objects (Rule II):**

- for any $r = (c_1, c_2) \in R$ and $o_2 \in \partial_{c_2}(O)$, we have that

$$|\{o_1 \in \partial_{c_1}(O) \mid o_1 \in \pi_{oomap}(o_2)\}| \in \sharp_{src}(r) \quad (3)$$

and³

- for any $r = (c_1, c_2) \in R$ and $o_1 \in \partial_{c_1}(O)$, we have that

$$|\{o_2 \in \partial_{c_2}(O) \mid o_2 \in \pi_{oomap}(o_1)\}| \in \sharp_{tar}(r) \quad (4)$$

Condition 1 and Condition 2 require that all relations between objects consist with the relationships of the class model, i.e., each object can only refer to objects of the class indicated by the class model. Condition 3 and Condition 4 imply that the number of objects related to each object should consist with the cardinality indicated by the class model.

Next, we illustrate how to check conformance on the data perspective based on Figure 3. The class model on the left side contains three classes (“order”, “order line” and “delivery”) and two relations with four cardinality constraints. The relation between “order” and “order line” indicates that each order corresponds to at least one order line and each order line corresponds to precisely one order. The relation between “delivery” and “order line” indicates that each delivery corresponds to at least one order line and each order line corresponds to precisely one delivery. The log contains two “order” objects ($o1$ and $o2$), three “order line” objects ($ol1$, $ol2$ and $ol3$) and two “delivery” objects ($d1$ and $d2$).

³ $\partial_{c_1}(O) = \{o \in O \mid \pi_{otyp}(o) = c_1\}$ are the objects corresponding to class $c_1 \in \mathcal{U}_C$.

After conformance checking on the data perspective, it is found that the object *ol2* has two corresponding “delivery” objects (*d1* and *d2*), thus violating the “1” annotation in terms of Rule II. The “order” object *o2* corresponds to the “delivery” object *d2*, indicating that they are deviations in terms of Rule I, since there is no relation between the “order” class and the “delivery” class in the model.

B. CONFORMANCE CHECKING ON BEHAVIORAL PERSPECTIVE

OCELs have no case notions to correlate events. In order to enable conformance checking on the behavioral perspective, we first use the data perspective as a bridge to correlate events, resulting in pattern instances. Then we compare the correlated instances with the activity model to detect deviations.

Definition 8 (Event Notations): Let $E \subseteq \mathcal{U}_E$ be a set of events ordered by \leq and related to activities through function π_{act} . For any event $e \in E$:

- $\leq_{\Delta}(E) = \{e' \in E \mid e' \leq e\}$ are the events *before* and *including* e .
- $\geq_{\Delta}(E) = \{e' \in E \mid e \leq e'\}$ are the events *after* and *including* e .
- $<_{\Delta}(E) = \{e' \in E \mid e' < e\}$ are the events *before* e .⁴
- $>_{\Delta}(E) = \{e' \in E \mid e < e'\}$ are the events *after* e .
- $\partial_a(E) = \{e' \in E \mid \pi_{act}(e') = a\}$ are the events corresponding to activity $a \in \mathcal{U}_A$.

Definition 9 (Event Correlation and Instances): Let L be an OCEL and con be a constraint. $a_{ref} = \pi_{ref}(con)$, $a_{tar} = \pi_{tar}(con)$ and $cr = crel(con)$. Function $extl \in \mathcal{U}_L \times \mathcal{U}_{Con} \rightarrow \mathbb{P}(E^*)$ correlates events in L for con and returns a set of instances (i.e., event sequences), such that $extl(L, con) = \{ins \in E^* \mid (\exists e_{ref} \in \partial_{a_{ref}}(E) : \partial_{set}(ins) = \{e_{ref}\} \cup E_{tar}) \wedge (\forall 1 \leq i < j \leq |ins| : ins_i < ins_j)\}$ where⁵

- $E_{tar} = \{e_{tar} \in \partial_{a_{tar}}(E) \mid \exists o \in \partial_c(O) : o \in \pi_{eomap}(e_{ref}) \cap \pi_{eomap}(e_{tar})\}$ if $cr = c \in C$, or
- $E_{tar} = \{e_{tar} \in \partial_{a_{tar}}(E) \mid \exists o_1 \in \pi_{eomap}(e_{ref}), o_2 \in \pi_{eomap}(e_{tar}) : o_2 \in \pi_{oomap}(o_1) \wedge \{\pi_{otyp}(o_1), \pi_{otyp}(o_2)\} = \{c_1, c_2\}\}$ if $cr = (c_1, c_2) \in R$.

For simplicity, we define $ins_{|con} = (before, after) = (|<_{e_{ref}}(E_{tar})|, |>_{e_{ref}}(E_{tar})|)$.

Function $extl$ correlates events in two ways. The first way is based on a triangle pattern, i.e., $crel(con) \in C$. More precisely, if two events refer to one common object, they are related, as shown in Figure 4. For instance, the event *co1* refers to two order line objects *ol1* and *ol2* while *ol1* is also referred to by *cs1* and both *ol1* and *ol2* are referred to by *cs2*. Therefore, *cs1* and *cs2* are the target events of the reference event *co1*, and the relating path is highlighted in red.

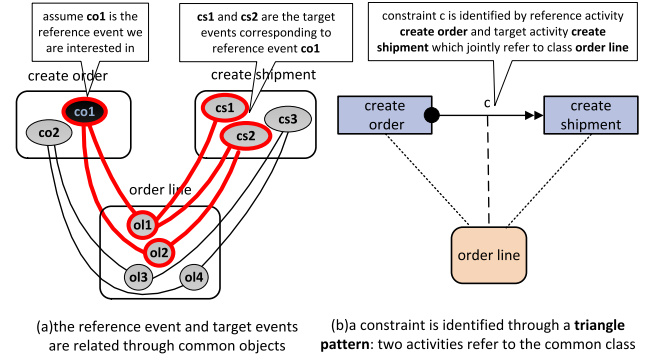


FIGURE 4. Given a reference event, we navigate to the target events through a triangle pattern.

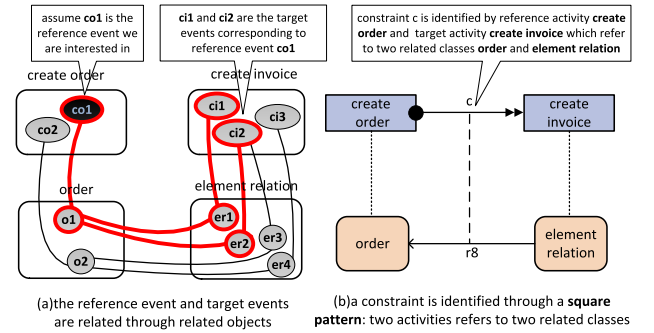


FIGURE 5. Given a reference event, we navigate to the target events through a square pattern.

The second way to relate events is by a square pattern, i.e., $crel(con) \in R$. More precisely, if two events refer to two related objects (which are connected by an object relation), they are related, as shown in Figure 5. For instance, the reference event *co1* refers to one order object *o1* related to *er1* and *er2* which are referred to by *ci1* and *ci2*, respectively. Therefore, *ci1* and *ci2* are the target events of *co1*.

Definition 10 (Conformance on Behavioral Perspective): Let L be an OCEL and M be an OCBC model. Event log L conforms to OCBC model M on the behavioral perspective if and only if the following rules are satisfied:

- **Each activity exists (Rule III):**

$$\{\pi_{act}(e) \mid e \in E\} \subseteq A \quad (5)$$

- **Each constraint is respected (Rule IV):** for each constraint $con \in Con$:

$$\forall ins \in extl(L, con) : ins_{|con} \in type(con) \quad (6)$$

Condition 5 means that all activities referred to by events exist in the activity model while Condition 6 means that all behavioral constraints are respected by correlated instances in the log.

Figure 6 shows how to check conformance on the behavioral perspective. The left part is a behavioral constraint of “unary-precedence” type between activities *a1* and *a2*. The constraint corresponds to a correlation pattern (*a2*, *a1*, *r*), i.e.,

⁴ $e' < e$ if and only if $e' \leq e$ and $e' \neq e$.

⁵ For a sequence σ , e.g., ins , σ_i refers to the i -th element of the sequence, $|\sigma|$ denotes the length of the sequence and $\partial_{set}(\sigma)$ converts the sequence into a set.

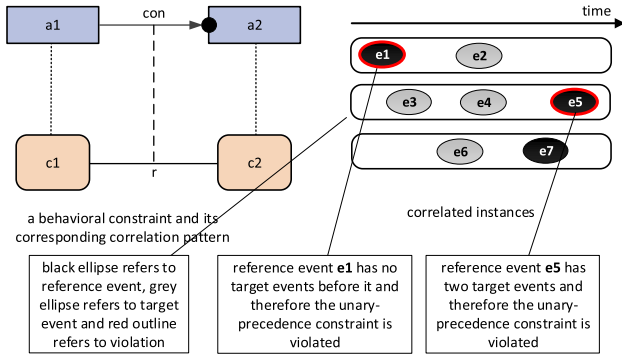


FIGURE 6. An illustration for checking conformance on the behavioral perspective (detecting violations of behavioral constraints).

$a2$ is the reference activity, $a1$ is the target activity and r is the intermediary. The right part presents three corresponding instances, in which $e1$, $e5$ and $e7$ are of activity $a2$, i.e., reference events, and the other events are target events. In each instance, the events are executed in the order indicated (from left to right).

The model in Figure 6 contains a constraint con where $type(con) = \{(before, after) \in \mathbb{N} \times \mathbb{N} \mid before = 1\}$, i.e., there should be precisely one target ($a1$) event preceding each reference ($a2$) event in each instance. Consider for example the third instance, in which $e7$ is the reference event and $e6$ is a target event occurring before $e7$. Hence, no problem is discovered for $e7$. In contrast, in the first instance there is no target events occurring before the reference event $e1$, which signals a violation of constraint con . For the reference event $e5$ in the second instance, there are two target events $e3$ and $e4$, which also violates con since the number “two” is not allowed by the constraint.

C. CONFORMANCE CHECKING ON INTERACTIONS

AOC relationships specify the interactions between the data perspective and the behavioral perspective. They indicate the allowed correspondences between events and objects. Next, we check the conformance on the interactions, i.e., compare the observed reference relations (between events and objects) in an OCEL with the AOC relationships in an OCBC model.

Definition 11 (Conformance on Interactions): Let L be an OCEL and M be an OCBC model. L conforms to M in terms of the interactions between data perspective and behavioral perspective if and only if the following rules are satisfied:

- **Each object has the right number of events (Rule V):** for any $aoc = (a, c) \in AOC$ and $o \in \partial_c(O)$:

$$|\{e \in \partial_a(E) \mid o \in \pi_{eomap}(e)\}| \in \sharp_A(a, c) \quad (7)$$

- **Each event refers to objects of related classes (Rule VI):** for any $e \in E$ and $o \in \pi_{eomap}(e)$:

$$(\pi_{act}(e), \pi_{otyp}(o)) \in AOC \quad (8)$$

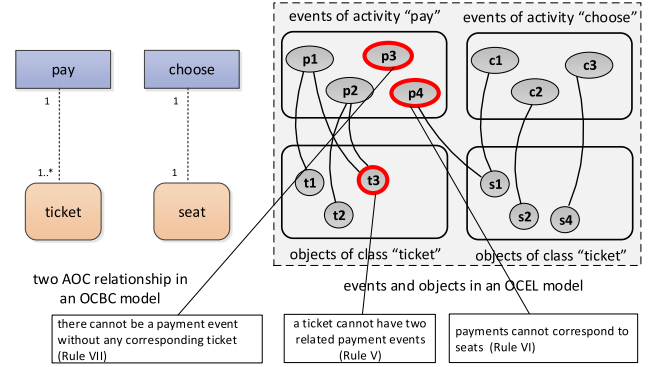


FIGURE 7. An illustration for checking conformance on interactions (detecting violations of cardinality constraints on AOC relationships between activities and classes).

- **Each event refers to the right number of objects (Rule VII):** for any $aoc = (a, c) \in AOC$ and $e \in \partial_a(E)$:

$$|\{o \in \pi_{eomap}(e) \mid \pi_{otyp}(o) = c\}| \in \sharp_{OC}(aoc) \quad (9)$$

Condition 7 indicates that for each object, the number of its related events correlated by an AOC relationship aoc satisfies the cardinality constraint, i.e., $\sharp_A(a, c)$. Condition 8 requires that the reference relations (between events and objects) should be consistent with AOC relationships. If an event e refers to an object o , there should exist an AOC relationship between the activity of e and the class of o . Otherwise, this rule is violated. Condition 9 implies that each event needs to have the right number of corresponding objects, correlated by an AOC relationship.

Figure 7 shows how to check conformance on interactions. The left part shows two AOC relations of an OCBC model. The relation between the activity “pay” and the class “ticket” indicates that each payment corresponds to at least one ticket and each ticket corresponds to precisely one payment. The relation between the activity “choose” and the class “seat” indicates that one person can choose only one seat and one seat can be chosen by only one person. The log contains four “pay” events ($p1$, $p2$, $p3$ and $p4$), three “choose” events ($c1$, $c2$ and $c3$), three “ticket” objects ($t1$, $t2$ and $t3$) and three “seat” objects ($s1$, $s2$ and $s3$).

After conformance checking on interactions, it is found that the object $t3$ has two corresponding “pay” events ($p1$ and $p2$), thus violating the “1” annotation in terms of Rule V. The “pay” event $p4$ corresponds to the “seat” object $s1$, indicating that it is a deviation in terms of Rule VI, since there is no AOC relation between the “pay” activity and the “seat” class in the model. Event $p3$ has no corresponding “ticket” objects, thus violating the “1..*” annotation according to Rule VII.

V. EXPERIMENTS

Next, we evaluate the OCBC conformance checking approach. More precisely, we first derive OCELS by simulating a designed model, and then inject some known

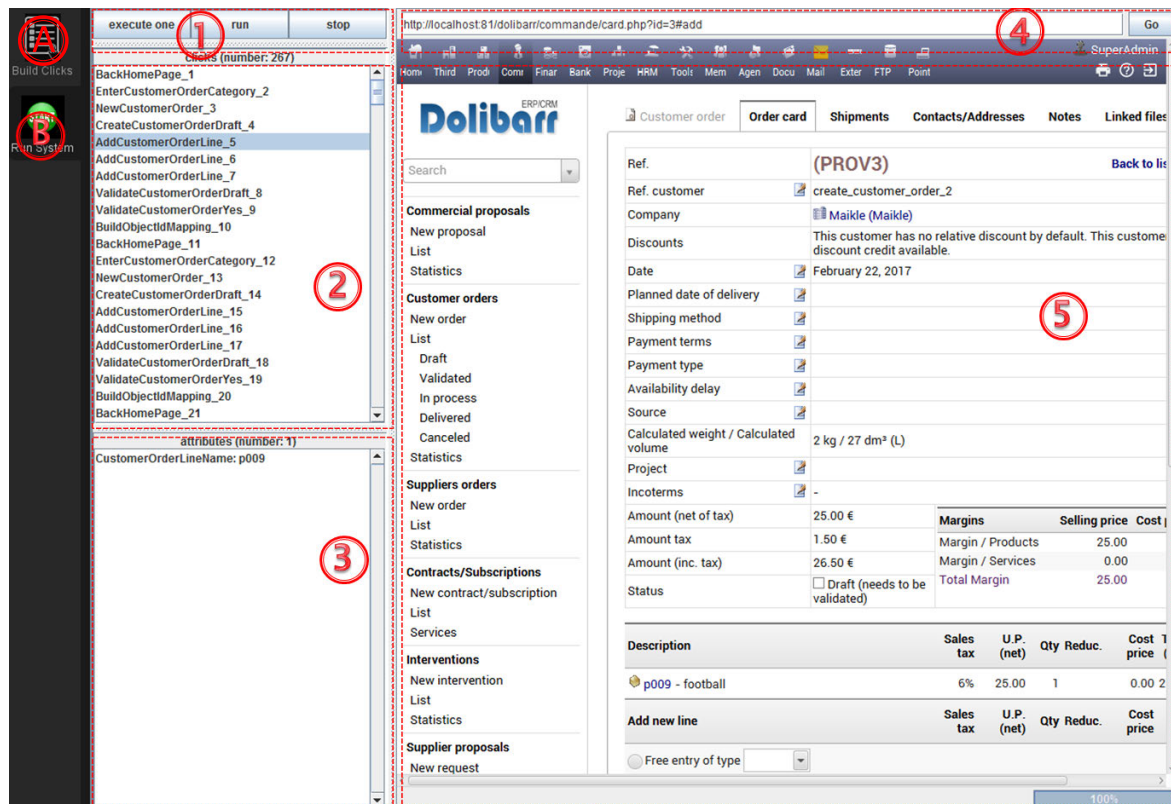


FIGURE 8. The “Run System” interface of the Data Generator.

deviations in these logs. Based on the reference model and generated logs, we check if our approach can detect these injected deviations. At last, our approach is compared with other conformance checking techniques.

A. GENERATING LOGS

OCELs are the input of our object-centric conformance checking approach. Such event logs are different from standard XES, MXML, and CSV log files in two respects: (i) there is no single case notion for the end-to-end process, and (ii) each event is related to one or multiple objects describing the entities of the process. This aligns well with the way that actual information systems work: data across different departments are stored in a database and transaction update the records in the database. Here we use *Dolibarr ERP/CRM* to illustrate the feasibility of the approach and the availability of the data assumed.

Based on an order-to-cash (OTC) process scenario, we firstly design a simulation model with CPN Tools (cpntools.org) [29], [34]. Then, a simulation log is generated by simulating complex process involving multiple interacting entities in the simulation model with CPN Tools. Next, we interpret the simulation log to automatically operate the Dolibarr ERP/CRM and populate the corresponding database using the tool as shown in Figure 8.⁶ For example, if an order

is created in the simulation and exported in the simulation log, it is also created in the real system. By running the simulation, the tables of Dolibarr get filled with information about orders, invoices, shipments, etc.

The tool in Figure 8 consists of two interfaces: “Build Clicks” (denoted as A) and “Run System” (denoted as B). More precisely, interface A builds an executable list of clicks by mapping click events in simulation logs onto buttons in interfaces, while interface B runs the system by triggering buttons based on the executable list. Interface B is the main component to control the execution of the information system. The basic idea is to embed the system into the interface, and control the execution of the system based on the click list. More precisely, panel 2 presents the click list while panel 3 shows the related attribute for the focused click. Panel 4 is used to fill in the website address and login. After providing the home page of the system, one can press “run” button in panel 1 to start running Dolibarr system. Panel 5 displays the state of Dolibarr system after each click, which is the same as operating Dolibarr in a browser. It is possible to suspend the execution using the “stop” button in panel 1 and resuming the execution using the “run” button. In order to investigate the details, one can press the “execute one” button to execute the click one by one.

After generating data in Dolibarr, we obtain tables such as *llx_commande* (order), *llx_commandedet* (order line),

⁶http://www.win.tue.nl/ocbc/softwares/data_generation.html

TABLE 1. Events of the input OCEL for conformance checking.

event id	activity	timestamp	related objects							
			order line	order	element relation	invoice	shipment	shipment line	payment line	payment
...
co251	create order	2023-7-20 08:15	{ol729,ol730}	{o251}	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
co252	create order	2023-7-20 11:30	{ol731,ol734}	{o252}	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
ci202	create invoice	2023-7-21 09:25	\emptyset	\emptyset	{er614}	{i202}	\emptyset	\emptyset	\emptyset	\emptyset
ci204	create invoice	2023-7-21 15:42	\emptyset	\emptyset	{er619}	{i204}	\emptyset	\emptyset	\emptyset	\emptyset
ci205	create invoice	2023-7-22 10:16	\emptyset	\emptyset	\emptyset	{i205}	\emptyset	\emptyset	\emptyset	\emptyset
cs365	create shipment	2023-7-23 09:28	\emptyset	\emptyset	\emptyset	\emptyset	{s365}	{sl425,sl426}	\emptyset	\emptyset
cs366	create shipment	2023-7-24 13:51	\emptyset	\emptyset	\emptyset	\emptyset	{s366}	{sl427}	\emptyset	\emptyset
cp181	create payment	2023-7-25 16:27	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{pl189}	{p181}
cp182	create payment	2023-7-26 09:53	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{pl191}	{p182}
...

TABLE 2. Objects of the input OCEL for conformance checking.

object id	class	customer	product	related objects						
				order line	order	element relation	invoice	shipment	shipment line	payment line
...
ol729	order line	...	computer	\emptyset	{o251}	\emptyset	\emptyset	\emptyset	{sl425}	\emptyset
ol730	order line	...	phone	\emptyset	{o251}	\emptyset	\emptyset	\emptyset	{sl427}	\emptyset
ol731	order line	...	cup	\emptyset	{o252}	\emptyset	\emptyset	\emptyset	{sl426}	\emptyset
ol734	order line	...	TV	\emptyset	{o252}	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
o251	order	Mike	...	{ol729,ol730}	\emptyset	{er619}	\emptyset	\emptyset	\emptyset	\emptyset
o252	order	Mike	...	{ol731,ol734}	\emptyset	{er614}	\emptyset	\emptyset	\emptyset	\emptyset
er614	element relation	\emptyset	{o252}	\emptyset	{i202}	\emptyset	\emptyset	\emptyset
er619	element relation	\emptyset	{o251}	\emptyset	{i204}	\emptyset	\emptyset	\emptyset
i202	invoice	Mike	...	\emptyset	\emptyset	{er614}	\emptyset	\emptyset	\emptyset	{pl189}
i204	invoice	Mike	...	\emptyset	\emptyset	{er619}	\emptyset	\emptyset	\emptyset	\emptyset
i205	invoice	Mike	...	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{pl191}
s365	shipment	Mike	...	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{sl425,sl426}	\emptyset
s366	shipment	Mike	...	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{sl427}	\emptyset
sl425	shipment line	...	computer	{ol729}	\emptyset	\emptyset	\emptyset	{s365}	\emptyset	\emptyset
sl426	shipment line	...	phone	{ol731}	\emptyset	\emptyset	\emptyset	{s365}	\emptyset	\emptyset
sl427	shipment line	...	cup	{ol730}	\emptyset	\emptyset	\emptyset	{s366}	\emptyset	\emptyset
pl189	payment line	\emptyset	\emptyset	\emptyset	{i202}	\emptyset	\emptyset	{p181}
pl191	payment line	\emptyset	\emptyset	\emptyset	{i205}	\emptyset	\emptyset	{p182}
p181	payment	Mike	...	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{pl189}	\emptyset
p182	payment	Mike	...	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	{pl191}	\emptyset
...

llx_expedition (shipment), llx_expeditiondet (shipment line), llx_facture (invoice), llx_facturedet (invoice line), llx_paiement (payment), llx_paiement_facture (payment line) and llx_element_element (element relation). Then we extract OCELs from the tables mentioned above. In order to conduct controlled experiments, we inject some deviations into the normal log and check whether the deviations that are injected can actually be discovered. Table 1 and Table 2 show the OCEL with deviations generated by simulating Dolibarr.

B. OBJECT-CENTRIC CONFORMANCE CHECKING

With the OCEL (in Table 1 and Table 2) and the OCBC model (in Figure 10) as input, we check the conformance to see if the inserted deviations can be identified. In our experiments, the injected deviations mentioned above can be totally detected. These deviations are highlighted in Figure 9 and explained as follows.

Like other modeling languages such as Petri nets, OCBC models support checking conformance on the behavioral

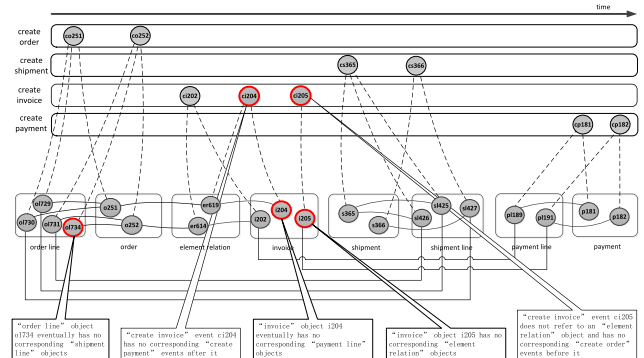


FIGURE 9. The conformance checking result in the log view.

perspective. In the normal scenario, each “create invoice” event is followed by at least one “create payment” event, indicated by the constraint *con22* in Figure 10. In the experiment, we found a deviation violating *con22*, i.e.,

a “create invoice” event (“ci204” in Figure 9) is never followed by corresponding “create payment” events.

A challenge for detecting behavioral deviations is how to detect implicit deviations. For instance, a “create order” event has corresponding “create shipment” events but does not have sufficient ones, i.e., order lines created by the “create order” event are not totally shipped to the corresponding customer. This implicit deviation is indeed violating our scenario but satisfying the behavioral constraint (i.e., *con6*, which requires a one-to-many relation between “create order” events and “create shipment” events). As OCBC models have a data perspective, it is possible to transform such implicit behavioral deviations onto the data perspective. For example, the deviation mentioned above can be interpreted as “some order lines have no corresponding shipment lines”, and be detected by checking the cardinality constraints on the class relationship *r9*. In the experiment, we found a deviating “order line” object *order_line734* (“ol734” in Figure 9) which has no corresponding “shipment line” objects.

In the Dolibarr system, when an invoice is created, it is generally linked to one or more existing orders. As a result, one or more element relations (showing the correspondence between the invoice and the orders) are created when a “create invoice” event happens, indicated by the cardinality “1..*” of the AOC relation *aoc3*. In reality, a possible deviating situation is that one forgets to link one invoice to any orders. In our experiment, we found an invoice (“ci205” in Figure 9) without any element relations, resulting in a deviation violating the constraints (i.e., “1..*” of *aoc3*) on the interactions.

Note that the deviations of different types depend on and impact each other. For instance, the deviation (related to the interactions) that the “create invoice” event *ci205* does not create an “element relation” object leads to two other additional deviations: object “i205” has no corresponding “element relation” objects (a deviation on the behavioral perspective) and event “ci205” cannot be correlated to a “create order” event before it (a deviation on the behavioral perspective).

Figure 10 shows the diagnosis result using a helicopter view (i.e., model view), by highlighting the violated constraints and cardinalities in the reference model. In summary, there are six constraints violated and highlighted in red, which corresponds to the detected deviations described in Figure 9. For instance, the annotation (1) indicates that some “create invoice” events are deviating since they are not followed by “create payment” events, and it corresponds to the deviating event *ci204*.

It is possible to dive into the highlighted relations. Consider for example the deviation (in Figure 9) that object “order_line734” has no corresponding “shipment_line” objects to understand how the model view display deviations. In Figure 10, the model view indicates the target cardinality (i.e., 1..*) of the class relationship between “order_line” and “shipment_line” is violated. By clicking this relationship, the

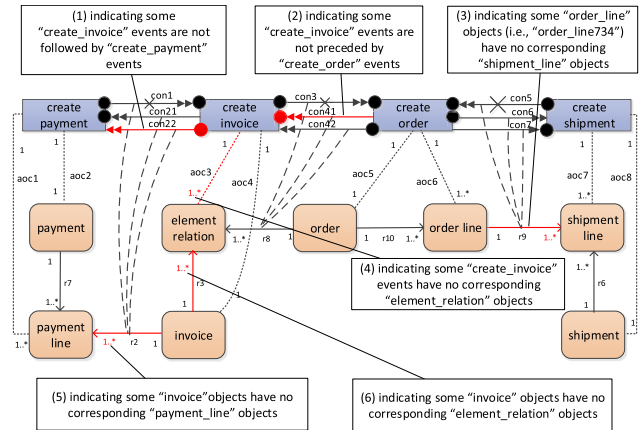


FIGURE 10. The conformance checking result in the model view.

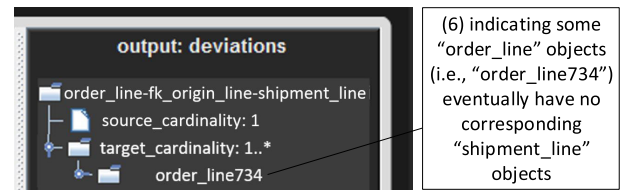


FIGURE 11. The conformance checking details.

“output:deviations” panel in Figure 11 displays the deviating object “order_line734” under its corresponding cardinality.

C. TRADITIONAL CONFORMANCE CHECKING

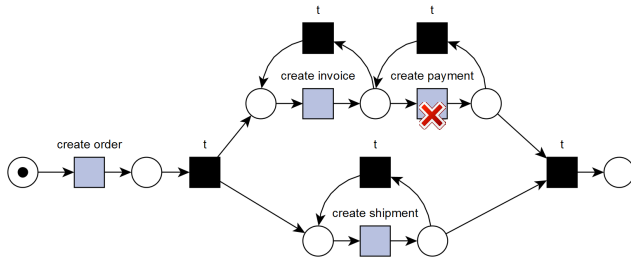
The diagnosis result derived by our object-centric approach shows that all inserted deviations can be found and presented through log and model views. In this part, we apply the traditional techniques to the same data and compare its results with the derived deviations using our approach.

The traditional conformance checking techniques (such as replay and alignment) often take an XES log and a Petri net as input, which assume a single case notions to correlate events. OCEs have different classes of objects, but have no case notion. Therefore, we select one class as the case notion to flatten OCEs into XES logs. Accordingly, the objects of the selected case class are called case objects, i.e., each case object can serve as a case id to integrate events into a case.

In order to correlate events to the case object, we first derive the so-called object chains based on the relations between objects in Table 2. For instance, “o251-er619-i204” is an object chain since “o251” is related to “er619” and “er619” is related to “i204” indicated by the sixth and ninth rows in Table 2. If an object chain starts with a case object, we call it a case object chain. For instance, “o251-er619-i204” is a case object chain when we select the “order” class as the case notion. Then, if an event has a related object which appears in a case object chain, the event will be added into the case corresponding to the case object. Accordingly, the fragment of the object chain, i.e., from the case object to the related object, serves as a path to correlate the event to the case object. For instance, event “ci204” is

TABLE 3. Events extracted from an OCEL with selecting order as the case notion.

case id	event id	activity	timestamp	associated to objects selected as case notion		
				related objects	paths	object chains
o251	co251	create order	2023-7-20 08:15	{ol729,ol730,o251}	{o251}	o251-er619-i204, o251-ol730-sl427-s366, o251-ol729-sl425-s365, o252-ol731-sl426-s365, o252-er614-i202-pl189-p181, o252-ol734
o251	ci204	create invoice	2023-7-21 15:42	{er619,i204}	{o251-er619}	
o251	cs365	create shipment	2023-7-23 09:28	{s365,sl425,sl426}	{o251-ol729-sl425}	
o251	cs366	create shipment	2023-7-24 13:51	{s366,sl427}	{o251-ol730-sl427}	
o252	co252	create order	2023-7-20 11:30	{ol731,ol734,o252}	{o252}	
o252	ci202	create invoice	2023-7-21 09:25	{er614,i202}	{o252-er614}	
o252	cs365	create shipment	2023-7-23 09:28	{s365,sl425,sl426}	{o252-ol731-sl426}	
o252	cp181	create payment	2023-7-25 16:27	{pl189,p181}	{o252-er614-i202-pl189}	

**FIGURE 12.** A Petri net to describe the OTC business process, including conformance checking result with respect to the log in Table 3. The cross symbol on “create payment” means that “create payment” events are mistakenly skipped in some cases.

related to object “er619”, which appears in the case object chain “o251-er619-i204”. Therefore, “ci204” is added into the case corresponding to “o251” and “o251-er619” is the correlation path in Table 3.

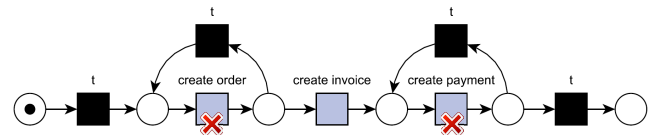
Based on the above method, we derive the XES log as shown in Table 3 with selecting the “order” class as the case notion. The XES log consists of two cases, as there exist two “order” objects *o251* and *o252*. The second case includes events *co252*, *ci202*, *cs365* and *cp181*, because these events are linked to *o252* by a path. Note that the events *ci205* and *cp182* are discarded when generating the XES log, because they are not linked to any “order” objects.

Figure 12 shows a designed Petri net to describe the OTC scenario. More precisely, after an order is created we have two branches. The top branch shows that multiple invoices can be created for the order and there exists a many-to-many relationship between invoices and payments, i.e., one invoice can be paid multiple times and one payment can cover multiple invoices. The two implicit transitions (i.e., *t*) describe this relationship. Independent from the top branch, the bottom one indicates that multiple shipments can be created to deliver order lines in the order. After all shipments and payments, the process ends.

After checking the conformance between the log in Table 3 and the model in Figure 12 using existing techniques, it is possible to detect some deviations on the behavioral perspective. For instance, the cross symbol on “create payment” in Figure 12 means that “create payment” events are mistakenly skipped in some cases. It indicates that the deviation that “create invoice” event *ci204* is not followed by a “create payment” event is detected by our approach.

However, the deviation that the “create invoice” event *ci205* has no corresponding “create order” event before it is not detected, as *ci205* is not included in the XES log in Table 3.

The diagnosis result with an assumed case notion of “order” on the whole process fails to detect all deviations, as a case notion only provides a view of the process from a particular angle. It is possible to detect the deviating “create invoice” event *ci205* by considering the sub-process related to “invoice”. Figure 13 shows the derived XES log based on the “invoice” case notion, and Figure 14 describes the sub-process related to “invoice”. Note that the “create shipment” activity is discarded in the sub-process because it is not related to “invoice”. After conformance checking, it is possible to detect more deviations on the behavioral perspective. The cross symbols on “create order” and “create payment” in Figure 14 means that “create order” or “create payment” events are mistakenly skipped in some cases. It indicates that the deviating event “create invoice” event *ci205* is detected, as *ci205* has no “create order” event before it.

**FIGURE 13.** The generated XES log assuming that “invoice” is the case notion.**FIGURE 14.** A Petri net to describe the sub-process related to “invoice” (i.e., the life-cycle of an invoice), including conformance checking result with respect to the log in Figure 13.

Based on some expertise, sometimes it is possible to detect the deviations on the data perspective indirectly. Consider for example the deviating object *ol734* (in Figure 9), which has no corresponding “shipment line” objects. By considering

“order line” as an artifact, we extract a corresponding XES log with four cases in Figure 15, and use the model in Figure 16 to describe its life-cycle.

After conformance checking, it is possible to detect more deviations on the behavioral perspective. The cross symbol on “create shipment” in Figure 16 means that “create shipment” events are mistakenly skipped in some cases. The result shows that the event *co252* has no following “create shipment” event in the case corresponding to *ol734*, indicating that *ol734* is deviating.

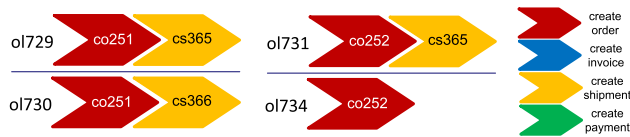


FIGURE 15. The generated XES log for the “order line” artifact.

The approach illustrated above indicates that the deviations on the data perspective can be detected if we can somehow transform them into deviations on the behavioral perspective, e.g., the deviating object *ol734*. However, if the deviation is related to objects which are not referred to by events, it is impossible to transform them into deviating events. For instance, assume that an “order” object does not have a corresponding “customer” object, which violates the class relationship *r5* in Figure 2. This deviation cannot be detected by the approach, because “customer” objects are not referred to by any events.

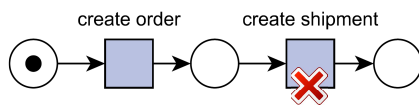


FIGURE 16. A Petri net to describe the life-cycle of the “order line” artifact, including conformance checking result with respect to the log in Figure 15.

In summary, traditional conformance checking techniques support detecting deviations on the behavioral perspective. They fail to efficiently detect deviations related to multiple instances and the data perspective, although it is possible to overcome some of the limitations by using some expertise such as choosing proper case notions, sub-processes or artifacts. In comparison, the OCBC conformance checking technique is more powerful in this situation. It can detect the deviations related to the data perspective and interactions in a straight-forward manner. Besides, the implicit deviations on the behavioral perspective can be detected when taking into consideration the data perspective.

VI. CONCLUSION

In this paper, we proposed techniques to check conformance based on OCBC models in terms of the behavioral perspective, data perspective and the interactions in between. In this way, we overcome the problems of existing approaches that instances are considered in isolation and constraints on the

data perspective are not taken into account. Hence, we can now detect and diagnose a range of conformance problems that would have remained undetected using conventional process model notations.

The conformance checking task is split into two parts in this paper. The first part is diagnosing the local conformance. We abstract a set of rules representing an OCBC model and detect eight types of conformance problems. The second task is to define three criteria, i.e., fitness, precision and generalization to quantify the conformance on the global level. At last, we evaluate our conformance checking approach based on the data generated in a real system Dolibarr by simulating the system.

REFERENCES

- [1] A. Rozinat and W. M. P. van der Aalst, “Conformance checking of processes based on monitoring real behavior,” *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, Mar. 2008.
- [2] G. Li and W. M. P. van der Aalst, “A framework for detecting deviations in complex event logs,” *Intell. Data Anal.*, vol. 21, no. 4, pp. 759–779, Aug. 2017.
- [3] W. M. P. van der Aalst, “Object-centric process mining: Dealing with divergence and convergence in event data,” in *Proc. Int. Conf. Softw. Eng. Formal Methods*. Oslo, Norway: Springer, Sep. 2019, pp. 3–25.
- [4] W. M. P. van der Aalst and A. Berti, “Discovering object-centric Petri nets,” *Fundamenta Informaticae*, vol. 175, nos. 1–4, pp. 1–40, 2020.
- [5] M. Zayoud, Y. Kotb, and S. Ionescu, “ β algorithm: A new probabilistic process learning approach for big data in healthcare,” *IEEE Access*, vol. 7, pp. 78842–78869, 2019.
- [6] A. K. A. de Medeiros, W. M. P. van der Aalst, and A. J. M. M. Weijters, “Quantifying process equivalence based on observed behavior,” *Data Knowl. Eng.*, vol. 64, no. 1, pp. 55–74, Jan. 2008.
- [7] A. Rozinat, “Process mining: Conformance and extension,” Ph.D. thesis, Dept. Math. Comput. Sci., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2010.
- [8] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, “Conformance checking using cost-based fitness analysis,” in *Proc. IEEE 15th Int. Enterprise Distrib. Object Comput. Conf.*, Aug. 2011, pp. 55–64.
- [9] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, “Towards robust conformance checking,” in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2010, pp. 122–133.
- [10] G. Li, “Mining based on object-centric behavioral constraint (OCBC) models,” Ph.D. thesis, Dept. Math. Comput. Sci., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2019.
- [11] G. Li, R. M. de Carvalho, and W. M. P. van der Aalst, “Automatic discovery of object-centric behavioral constraint models,” in *Proc. Int. Conf. Bus. Inf. Syst.* Cham, Switzerland: Springer, 2017, pp. 43–58.
- [12] W. M. P. van der Aalst, G. Li, and M. Montali, “Object-centric behavioral constraints,” 2017, *arXiv:1703.05740*.
- [13] A. F. Ghahfarokhi, G. Park, A. Berti, and W. M. P. van der Aalst, “OCEL: A standard for object-centric event logs,” in *New Trends in Database and Information Systems*. Cham, Switzerland: Springer, 2021, pp. 169–175.
- [14] A. Adriansyah, “Aligning observed and modeled behavior,” Ph.D. thesis, Dept. Math. Comput. Sci., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2014.
- [15] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, “Measuring precision of modeled behavior,” *Inf. Syst. e-Bus. Manage.*, vol. 13, no. 1, pp. 37–67, Feb. 2015.
- [16] J. Muñoz-Gama and J. Carmona, “A fresh look at precision in process conformance,” in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2010, pp. 211–226.
- [17] J. Muñoz-Gama and J. Carmona, “Enhancing precision in process conformance: Stability, confidence and severity,” in *Proc. IEEE Symp. Comput. Intell. Data Mining (CIDM)*, Apr. 2011, pp. 184–191.
- [18] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, “Alignment based precision checking,” in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2012, pp. 137–149.

- [19] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, Mar. 2012.
- [20] D. Fahland, M. de Leoni, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking of interacting processes with overlapping instances," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2011, pp. 345–361.
- [21] D. Fahland, M. de Leoni, B. F. van Dongen, and W. M. P. van der Aalst, "Behavioral conformance of artifact-centric process models," in *Proc. Int. Conf. Bus. Inf. Syst.* Cham, Switzerland: Springer, 2011, pp. 37–49.
- [22] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2012, pp. 82–97.
- [23] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Syst.*, vol. 47, pp. 258–277, Jan. 2015.
- [24] D. Zhao, W. Gaaloul, W. Zhang, C. Zhu, and Z. Zhou, "Formal verification of temporal constraints for mobile service-based business process models," *IEEE Access*, vol. 6, pp. 59843–59852, 2018.
- [25] R. Ben Halima, I. Zouaghi, S. Kallel, W. Gaaloul, and M. Jmaiel, "Formal verification of temporal constraints and allocated cloud resources in business processes," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2018, pp. 952–959.
- [26] M. de Leoni, W. M. P. van der Aalst, and B. F. van Dongen, "Data-and resource-aware conformance checking of business processes," in *Proc. Int. Conf. Bus. Inf. Syst.* Cham, Switzerland: Springer, 2012, pp. 48–59.
- [27] F. Mannhardt, "Multi-perspective process mining," Ph.D. thesis, Dept. Math. Comput. Sci., Univ. Michigan, Eindhoven, The Netherlands, 2018.
- [28] J. C. Carrasquel, K. Mecheraoui, and I. A. Lomazova, "Checking conformance between colored Petri nets and event logs," in *Proc. Int. Conf. Anal. Images, Social Netw. Texts.* Skolkovo, Russia: Springer, Oct. 2020, pp. 435–452.
- [29] K. Jensen, "Coloured Petri nets," in *Advances in Petri Nets 1986, Part I on Petri Nets: Central Models and Their Properties.* Berlin, Germany: Springer, 1987, pp. 248–299.
- [30] W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: Models, Methods, and Systems.* Cambridge, MA, USA: MIT Press, 2002.
- [31] G. Li, R. Medeiros de Carvalho, and W. M. P. van der Aalst, "Configurable event correlation for process discovery from object-centric event data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2018, pp. 203–210.
- [32] G. Li, E. G. L. de Murillas, R. M. de Carvalho, and W. M. P. van der Aalst, "Extracting object-centric event logs to support process mining on databases," in *Information Systems in the Big Data Era.* Cham, Switzerland: Springer, 2018, pp. 182–199.
- [33] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Comput. Sci. Res. Develop.*, vol. 23, no. 2, pp. 99–113, May 2009.
- [34] W. M. P. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach.* Cambridge, MA, USA: MIT Press, 2011.



BAOXIN XIU received the bachelor's degree in applied mathematics, in 2000, and the Ph.D. degree in management science and engineering, in 2006. He is currently a Professor with the School of Systems Science and Engineering, Sun Yat-sen University. His research interests include complex systems and organization theory.



GUANGMING LI received the B.E. degree in electrical engineering from the Harbin University of Technology, in 2012, and the M.Sc. degree in management science and engineering and the Ph.D. degree from the Section of Information System (IS), Department of Mathematics and Computer Science, Eindhoven University of Technology, in 2014 and 2019, respectively. His research interests include process mining, business process modeling, and object-centric information systems.

...