

## RESEARCH ARTICLE

# Quantifying Conformance Between Object-Centric Event Logs and Models

BAOXIN XIU<sup>1</sup> AND GUANGMING LI<sup>2</sup> <sup>1</sup>School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou 510000, China<sup>2</sup>PLA No. 31307, Chengdu 610000, China

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

**ABSTRACT** Conformance checking techniques can reveal commonalities and discrepancies between the observed behavior and the modeled behavior, by relating and comparing events in the event log to activities in the process model. Existing techniques quantify conformance based on mainstream models like Petri nets, which are process-centric and assume a case notion. However, the information systems which are widely used in current organizations are object-centric and do not have a case notion. As a result, existing techniques fail to accurately quantify the conformance for object-centric data. Therefore, in this paper, we propose a new approach to quantify the conformance between object-centric models and object-centric logs, and redefine metrics such as fitness and precision. At last, our approach is verified to be effective.

**INDEX TERMS** Process mining, conformance quantifying, metrics, object-centric event logs, enterprise resource planning systems.

## I. INTRODUCTION

Process models are widely employed to indicate how their business processes should be executed. However, the predefined processes are often violated in some situations, leading to conformance problems between predefined processes and real executions [1]. For example, people actually using SAP R/3 may deviate from these reference models [2]. For companies, it is necessary to quantify the conformance between the observed behavior and the modeled behavior, and remove the violations corresponding to bad situations. For instance, in financial systems, it is desirable to keep the actual procedure consistent with the model.


The existing conformance checking techniques often employ process-centric logs and models which consider process instances in isolation due to the assumption of a single case notion [3], [4], [5], [6], [7], [8]. However, enterprise resource planning (ERP) and customer relationship management (CRM) systems which are widely used in current organizations are object-centric and do not assume a case notion. Accordingly, when applying existing techniques on these systems, traditional logs and models suffer convergence

and divergence problems, since they are flattened and can only focus on a particular perspective of the process [9], [10], [11]. As a result, conformance checking fail to detect deviations and the quantifying result is not accurate.

Fortunately, object-centric process mining have been proposed to tackle the problem. Extensible object-centric (XOC) event logs, are employed in [12] to check and quantify conformance. These conformance checking techniques are not widely used since XOC logs suffer complexity and performance problems. Reference [13] checks conformance between object-centric logs and models based on seven rules, without proposing metrics to quantify conformance.

Fitness, simplicity, precision, and generalization are used to evaluate the quality of a model discovered from a log in traditional process mining. Actually, these criteria can also reveal the conformance between the log and the discovered model, since a discovered model has good quality if it conforms to the log. In this paper, we redefine fitness and precision in an object-centric way to quantify the conformance. Different from traditional process mining, a global case notion is not assumed for the whole process in the context of an object-centric model and an object-centric log.

This paper is organized as follows. Section V presents the idea to quantify the conformance and provides a solution

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

to connect a log and a model. In order to quantify the conformance, we redefine existing criteria, i.e., fitness and precision, and propose approaches to compute them in an object-centric manner in Section VI and VII, respectively. In Section VIII, we evaluate our approach and compare it with other conformance checking techniques. Section IX concludes this paper.

## II. NOVELTY OF THE RESEARCH

The conformance problems between predefined processes and real executions often happen in companies due to the flexible operation environment. These problems will weaken the internal control and lead to inefficiencies in companies. Some undesirable deviations even suggest that specific rules enforced by law or company policies are violated. Conformance checking is related to business alignment and auditing and can find commonalities and discrepancies between the modeled behavior and the observed behavior. This is done to check whether business processes are executed within certain boundaries set by managers, governments, and other stakeholders. By improving the alignment of information systems, business processes, and the organization and diagnosing discrepancies, new insights can be gathered showing how to solve the problems of fraud, malpractice, risks, and inefficiencies and improve the performance by maximizing the benefits.

Existing techniques quantify conformance based on mainstream models which typically consider process instances in isolation (ignoring interactions among them) and are more focused on the behavioral perspective of processes. However, the information systems which are widely used in current organizations are object-centric and do not have a case notion. As a result, existing techniques fail to accurately quantify the conformance for object-centric data.

Therefore, in this paper, we redefine metrics such as fitness and precision to quantify the conformance between object-centric models and object-centric logs. These metrics are different from traditional process mining since they do not assume a global case notion for the whole process. More precisely, in an object-centric model [12], one behavioral constraint specifies a restriction on events in a scope (i.e., an instance rather than a case) identified by a named correlation pattern [12]. A named variant matrix is served as a bridge to connect the log and the model [12]. More precisely, we map the instances onto the variant matrix to identify the observed variants, and map the constraints onto the variant matrix to identify the allowed ones. Then the criteria are computed based on the relation between observed and allowed variants.

A major contribution of our approach is that instances are not considered in isolation and cardinality constraints in the data/object model are taken into account. Therefore, when applying our conformance checking techniques on the object-centric systems, the diagnosis results can cover both the behavioral and data perspectives, and deal deviations related to the interactions. Hence, we can now detect and

diagnose a range of conformance problems that would have remained undetected using existing approaches. Besides, compared with token-based replay approaches which tend to achieve too high fitness values when event logs contain many deviations, our approach can quantify a range of conformance problems more precisely, since our metrics does not assume a global case notion and can overcome famous convergence and divergence problems.

## III. RELATED WORK

Various conformance checking techniques are proposed to quantify the conformance problems. In this section, we discuss the literature related to approaches to quantify the conformance between a log and a model.

In traditional process mining, the replay techniques are used to compute fitness between a Petri net and an extensible event stream (XES) log [2], [14], [15]. By replaying the log on the model, i.e., replaying each case to the places and transitions, four counters are employed to count produced tokens ( $p$ ), consumed tokens ( $c$ ), remaining tokens ( $r$ ) and missing tokens ( $m$ ). For instance, if an activity in the event log is not enabled, then a missing token is added. Based on these four numbers, a typical application is to compute fitness based on the numbers. More precisely, after replaying a trace  $\sigma$  in a log on top of a model  $N$ , the fitness of a trace  $\sigma$  is calculated as  $fitness(\sigma, N) = \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p})$ . The same approach can be used to analyze the fitness of a log consisting of many cases: simply taking the sums of all produced, consumed, missing, and remaining tokens, and applying the same formula. Reference [16] proposes an improved token-based replay approach that is much faster and scalable than existing methods. Moreover, the approach provides more accurate diagnostics that avoid known problems and help to pinpoint compliance problems. Inspired by the object-centric paradigm [17], [18], [19], [20] presents a replay-based conformance checking method based on a class of colored Petri nets (CPNs) which are frequently used to describe systems centered on the end-to-end processing of distinguishable objects. It can check conformance in terms of interaction between multiple process instances in a system, which addresses the limitation that the majority of existing methods focus on checking isolated process instances. Reference [21] introduces a notion for the precision and fitness of an object-centric Petri net with respect to an object-centric event log, which is able to handle multiple case notions, their dependencies and their interactions.

However, token-based replay can only be implemented on Petri nets. Besides, for event logs with many deviations, this approach tends to achieve too high fitness values, since Petri net is flooded with tokens and allows for too much behavior. In order to overcome these problems faced by token-based replay, alignment techniques are proposed and become the state-of-the-art techniques. The alignment techniques [7], [8], [22], [23] compute fitness based on the costs, assigning to

the misalignment between model moves (not observed in the log) and log moves (impossible in the model). For instance, if each model path is the same as the log path (i.e., there are no model moves and log moves), the cost is zero, which means the match is perfect. Reference [24] presents a novel approach to incrementally calculate prefix-alignments, paving the way for real-time online conformance checking more efficiently. Reference [25] defines infix/postfix alignments, and proposes approaches to compute them, without assuming complete process executions covering the entire process from start to finish or prefixes of process executions.

Reference [3] and [4] employ artifact-centric models expressed in terms of procllets to check conformance. These papers show that process instances cannot be considered in isolation as instances in artifact-centric processes may overlap and interact with each other. This complicates conformance checking but the problem can be decomposed into a set of smaller problems, that can be analyzed using conventional conformance checking techniques. In contrast to most existing approaches, [26] incorporates data constraints into the Petri nets by relying on the expressive power of an artifact-centric specification, thus achieving conformance results which are more precise. These artifact-centric conformance checking techniques do not relate control-flow to some overall data model.

These techniques more focus on quantifying the level of conformance in terms of fitness. However, there are also techniques for computing other quality dimensions such as simplicity and precision. In [27] and [28], the non-fitting parts are simply ignored, i.e., only a fraction of the event log can be used for computing precision, resulting in unreliable precision measurements. Reference [29] can achieve better precision by aligning the event log and the model in case of deviations. Reference [21] can calculate precision for multiple case notions based on an object-centric Petri net. Reference [30] measures the generalization criterion when the model is “overfitting” (i.e., the model explains the particular sample log, but it is unlikely that another sample log of the same process can be explained well by the current model).

## IV. PRELIMINARY

### A. OBJECT-CENTRIC EVENT LOGS

*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,
- $\pi_{evmap} \in E \rightarrow (\mathcal{U}_{Attr} \dashv \mathcal{U}_{Val})$  maps events onto a partial function assigning values to some attributes,<sup>1</sup>
- $\pi_{eomap} \in E \rightarrow \mathcal{P}(O)$  associates events to objects,
- $\pi_{otyp} \in O \rightarrow \mathcal{U}_C$  maps objects onto object types (or classes),
- $\pi_{ovmap} \in O \rightarrow (\mathcal{U}_{Attr} \dashv \mathcal{U}_{Val})$  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.<sup>2</sup>

$\mathcal{U}_L$  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 3 and Table 4 briefly present an object-centric event log [31], [32]. Table 3 represents the event records, where each row corresponds to a distinct event. Table 4 represents the relevant information of objects in the information systems.

### B. OBJECT-CENTRIC CONSTRAINT BEHAVIORAL MODEL

An OCBC model combines data/object modeling techniques and a declarative process modeling language. 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

<sup>1</sup> $f \in X \dashv Y$  is a partial function with domain  $dom(f) \subseteq X$ .

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

- $\#_{tar} \in R \rightarrow \mathcal{U}_{Card}$  gives the target cardinality of a relationship (i.e.,  $\#_{tar}(r)$  gives the cardinality on the  $c_2$  side for  $r = (c_1, c_2) \in R$ ).

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

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

- $A \subseteq \mathcal{U}_A$  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
- $type \in Con \rightarrow \mathcal{U}_T$  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.

- $\#_A \in AOC \rightarrow \mathcal{U}_{Card}$  gives the source cardinality of an AOC relationship (activity side), and
- $\#_{OC} \in AOC \rightarrow \mathcal{U}_{Card}$  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  $OCBCM = (Clam, ActM, AOC, \#_A, \#_{OC}, crel)$ , where

- $Clam = (C, R, \#_{src}, \#_{tar})$  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  $\#_A$  and  $\#_{OC}$  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 10 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.

### C. EVENT CORRELATION

OCEs 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 7 (Event Notations):** Let  $E \subseteq \mathcal{U}_E$  be a set of events ordered by  $\leq$  and related to activities through function  $\pi_{act}$ . For any event  $e \in E$ :

- $\triangleleft_e(E) = \{e' \in E \mid e' \leq e\}$  are the events *before and including*  $e$ .
- $\triangleright_e(E) = \{e' \in E \mid e \leq e'\}$  are the events *after and including*  $e$ .
- $\triangleleft_e^<(E) = \{e' \in E \mid e' < e\}$  are the events *before*  $e$ .<sup>3</sup>
- $\triangleright_e^<(E) = \{e' \in E \mid e < e'\}$  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 8 (Correlation Pattern):** A correlation pattern  $P$  is a tuple  $(a_{ref}, a_{tar}, cr)$  where  $a_{ref}$  and  $a_{tar}$  are two activities, and  $cr$  is a class or class relationship in an OCBC model.

A correlation pattern  $P = (a_{ref}, a_{tar}, cr)$  consists of two activities and a class or class relationship (which serves as a “bridge” to connect these two activities) in an OCBC model [12]. For instance, as shown in Figure 1, these two activities (“create invoice” and “create order”) and the class relationship (between “invoice” and “order”) forms a correlation pattern.

**Definition 9 (Event Correlation and Instances):** Let  $L$  be an OCEL and  $P = (a_{ref}, a_{tar}, cr)$  be a correlation pattern. Function  $extI \in \mathcal{U}_L \times \mathcal{U}_P \rightarrow \mathbb{P}(E^*)$  correlates events in  $L$  for  $P$  and returns a set of instances (i.e., event sequences), such that  $extI(L, P) = \{ins \in E^* \mid (\exists e_{ref} \in \partial_{a_{ref}}(E) : \partial_{set}(ins) = \{e_{ref}\} \cup E_{tar}) \wedge (\forall 1 \leq i < j \leq |ins| : ins_i < ins_j)\}$  where<sup>4</sup>

- $E_{tar} = \{e_{tar} \in \partial_{a_{tar}}(E) \mid \exists o \in \partial_c(O) : o \in \pi_{eomap}(e_{ref}) \cap \pi_{eomap}(e_{tar})\}$  if  $cr = c \in C$ , or

<sup>3</sup>  $e' < e$  if and only if  $e' \leq e$  and  $e' \neq e$ .

<sup>4</sup> For a sequence  $\sigma$ , e.g.,  $ins$ ,  $\sigma_i$  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.

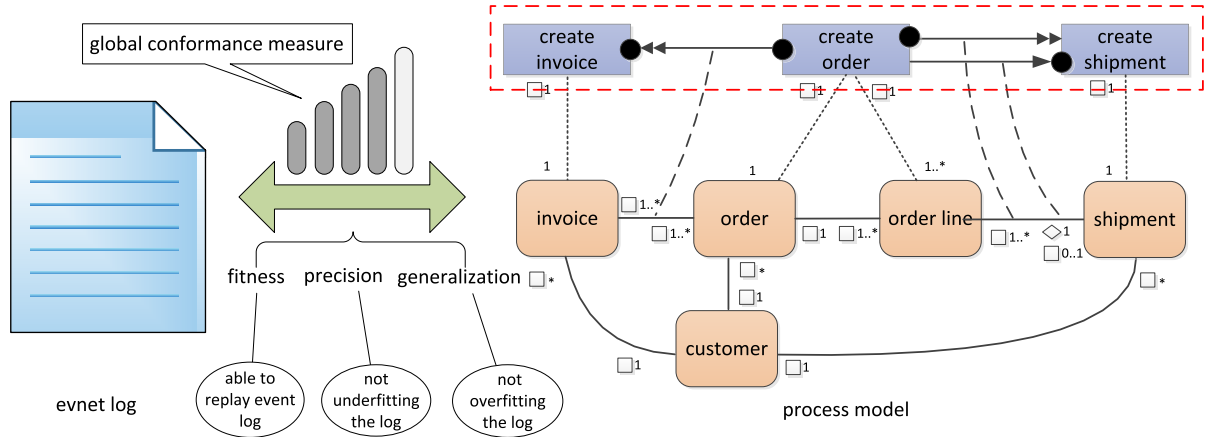


FIGURE 1. Quantifying the conformance on the behavioral perspective through three criteria: fitness, precision and generalization.

$$E_{tar} = \{e_{tar} \in \partial_{a_{tar}}(E) \mid \exists o_1 \in \pi_{eomap}(e_{ref}), o_2 \in \pi_{eomap}(e_{tar}) : o_2 \in \pi_{oomap}(o_1) \wedge \{\pi_{otyp}(o_1), \pi_{otyp}(o_2)\} = \{c_1, c_2\}\} \text{ if } cr = (c_1, c_2) \in R.$$

For simplicity, we define  $ins_{LP} = (before, after) = (|\triangleleft_{e_{ref}}(E_{tar})|, |\triangleright_{e_{ref}}(E_{tar})|)$ .

Function  $extI$  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. 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.

## V. THE FRAMEWORK FOR QUANTIFYING CONFORMANCE

### A. BASIC IDEA OF CRITERIA

Quantifying the conformance between a log and a model is difficult, since it is characterized by many dimensions. Fitness and precision are used to evaluate the quality of a model discovered from a log. Actually, these criteria can also reveal the conformance between the log and the discovered model, since a discovered model has good quality if it conforms to the log. Therefore, in this paper, we redefine fitness and precision to quantify the conformance, as shown in Figure 1. Note that these criteria only focus on the behavioral perspective, which are explained as follows:

- fitness: the model should allow for the behavior seen in the event log,
- precision: the model should not allow for behavior completely unrelated to what was seen in the event log, and
- generalization: the model should generalize the behavior seen in the event log.

Unlike traditional process mining, we do not assume a global case notion for the whole process in the context of an OCBC model and an object-centric event log (OCEL) [13]. Each behavioral constraint in an OCBC model corresponds to a correlation pattern and specifies a restriction on events in a scope (rather than a case) identified by the pattern. Since a behavioral constraint is defined in the context of a correlation

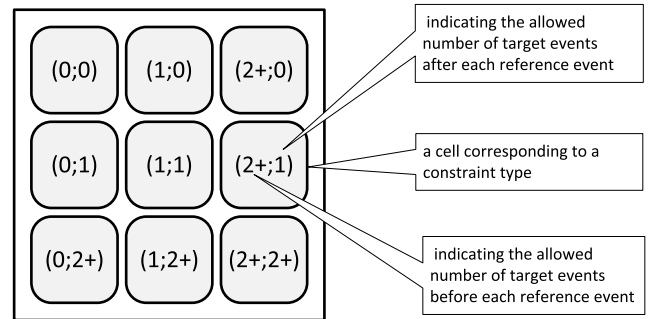


FIGURE 2. A variant matrix is represented by a grid consisting of 3 x 3 cells.

pattern, constraints corresponding to different patterns are independent. Because of this, it is not necessary to check an OCEL log on the whole OCBC model (with all activities and constraints). We only need to *check conformance pattern by pattern*.

In this paper, we define and compute criteria on the pattern level. More precisely, based on a correlation pattern, we correlate events in the log to derive a set of pattern instances and extract all behavioral constraints corresponding to the pattern from the model. The criteria on the pattern level are computed for each pattern, which can be merged as criteria on the model level if needed.

### B. CONNECTING EVENT LOG AND PROCESS MODEL

The conformance between a model and a log means how much the observed behavior complies with the allowed behavior. Therefore, we have to connect the model to the log to quantify the conformance.

**Definition 10 (Variant Matrix):** A variant matrix  $V_{CT}$  is a set of nine disjoint constraint types which incorporate all the possible relations between a reference event and its target events.  $V_{CT} = \{(0; 0), (1; 0), (0; 1), (1; 1), (2+; 0), (0; 2+), (2+; 1), (1; 2+), (2+; 2+)\} \subseteq \mathcal{U}_{CT}$ . Each constraint type in the set is called a variant, e.g.,  $(1; 1) \in V_{CT}$  is a variant.

In the variant matrix, each variant essentially is a constraint type, e.g.,  $(2+; 0) = \{(before, after) \in \mathbb{N} \times \mathbb{N} \mid before \geq 2\}$ .

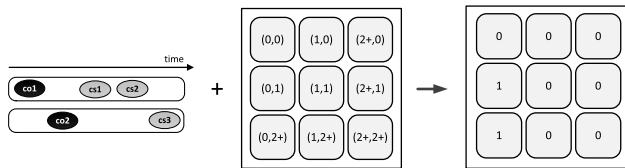
The variant matrix  $V_{CT}$  in Figure 2 can serve as a bridge to connect instances to constraints. More precisely, we map the instances onto the variant matrix to identify the observed variants, and map the constraints onto the variant matrix to identify the allowed ones. Then the criteria are computed based on the relation between observed and allowed variants. Next, we explain how to derive the observed variants and their frequencies.

Among the instances correlated by a candidate pattern, if the relation between a reference event and its target events in some instance satisfies the semantics of a variant, we say that the instance corresponds to the variant and the variant is observed once in the log. Based on this idea, a function  $freV$  is defined to compute how many times each variant is observed in a log corresponding to a pattern.

**Definition 11 (Computing Variant Frequency):** Let  $L$  be an OCEL and  $P$  be a correlation pattern. Function  $freV \in \mathcal{U}_L \times \mathcal{U}_P \times V_{CT} \rightarrow \mathbb{N}$  returns the frequency that a variant is observed in a log corresponding to a pattern such that  $freV(L, P, v) = |\{ins \mid ins \in extI(L, P) \wedge ins|_P \in v\}|$ .

For convenience, we define the following shorthand.  $freV_{\%}(L, P, v) = \frac{freV(L, P, v)}{\sum_{v \in V_{CT}} freV(L, P, v)}$  provides the ratio of a variant.

More precisely, the frequency of a variant is equal to the number of instances corresponding to the variant. For instance, assume that we have two instances  $\{\langle co1, cs1, cs2 \rangle, \langle co2, cs3 \rangle\}$  where  $co1$  and  $co2$  are reference events in Figure 3. For the first instance, there are zero and two target events before and after the reference events, respectively, i.e.,  $\langle co1, cs1, cs2 \rangle|_P = (0, 2) \in (0; 2+)$ . Similarly  $\langle co2, cs3 \rangle|_P = (0, 1) \in (0; 1)$ . Therefore  $freV(L, P, (0; 2+)) = 1$  and  $freV(L, P, (0; 1)) = 1$ .



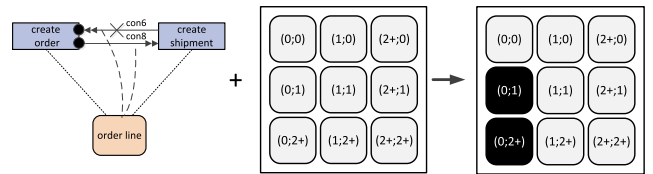
**FIGURE 3. Function  $freV$  mapping instances (correlated by a correlation pattern) onto the variant matrix  $V_{CT}$  to show observed behavior, resulting in a frequency matrix. For instance, the first one of the instances (on the left side) only has two target events  $cs1$  and  $cs2$  after the reference event  $co1$ . This relation satisfies the semantics of the cell  $(0; 2+)$  in the variant matrix (in the middle). Since only one of the instances has such relation, the value in the cell corresponding to  $(0; 2+)$  in the frequency matrix (on the right side) is 1.**

In an OCBC model, each behavioral constraint corresponds to a constraint type. Consider for example the constraint  $con8$  in in Figure 4.  $type(con8) = \{(before, after) \in \mathbb{N} \times \mathbb{N} \mid after \geq 1\}$ . Therefore, constraints can be related to the variant matrix in terms of constraint types. Next, we define a function to map constraints (corresponding to a correlation pattern) onto a variant matrix.

**Definition 12 (Allowed Variants by Model):** Let  $M = (ClaM, ActM, AOC, \#_A^{\square}, \#_A^{\diamond}, \#_{OC}, crel)$  be an OCBC model where  $ActM = (A, Con, \pi_{ref}, \pi_{tar}, type)$  is an activity model, and  $P = (a_{ref}, a_{tar}, cr)$  is a correlation pattern.

Function  $posV \in \mathcal{U}_{OCBCM} \times \mathcal{U}_P \rightarrow \mathbb{P}(V_{CT})$  returns the variants allowed by a model corresponding to a correlation pattern such that  $posV(M, P) = \{v \in V_{CT} \mid \forall con' \in Con_P : v \subseteq type(con')\}$  where  $Con_P = \{con \in Con \mid \pi_{ref}(con) = a_{ref} \wedge \pi_{tar}(con) = a_{tar} \wedge crel(con) = cr\}$ .

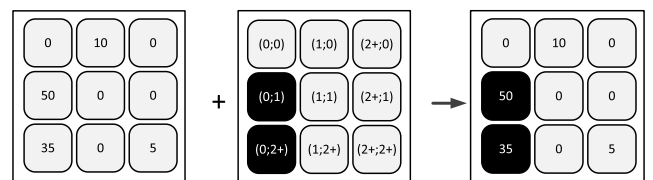
In terms of a correlation pattern  $P$ , function  $posV$  maps the constraints corresponding to  $P$  from an OCBC model  $M$  onto the variant matrix, resulting in a colored matrix. In the colored matrix, the black variants represent the behavior allowed by the model. Assume that  $con6$  and  $con8$  in Figure 4 are all constraints corresponding to the pattern  $P = (create\ order, create\ shipment, order\ line)$  from the OCBC model  $M$ . Since  $con6$  is of the *non-precedence* constraint type and  $con8$  is of the *response* constraint type,  $type(con6) \cap type(con8) = \{(before, after) \in \mathbb{N} \times \mathbb{N} \mid before = 0 \wedge after \geq 1\}$ . After checking all variants in the matrix, we get  $(0; 1) \subseteq (type(con6) \cap type(con8))$  and  $(0; 2+) \subseteq (type(con6) \cap type(con8))$ . Therefore,  $posV(M, P) = \{(0; 1), (0; 2+)\}$ .



**FIGURE 4. Function  $posV$  mapping constraints (for a correlation pattern) onto the variant matrix to show allowed behavior, resulting in a colored matrix.**

Traditional process mining techniques connect a log to a model by replaying each case in the log onto the model. Since OCELS and OCBC models do not assume a case notion, the variant matrix is employed to realize the connection. The functions proposed above can map an OCEL on the variant matrix (resulting in a frequency matrix to show the observed variants) and to map an OCBC model onto the variant matrix (resulting in a colored matrix to show the allowed variants). Next, the OCEL is connected to the OCBC model by overlapping the frequency matrix and the colored matrix.

**Definition 13 (Connecting OCEL to OCBC Model):** Function  $posV$  indicates a colored matrix which represents the allowed behavior, and function  $freV$  indicates a frequency matrix which represents the observed behavior. By overlapping and aligning these two matrices, an overlapped matrix is generated, which connects an OCEL to an OCBC model



**FIGURE 5. A log is connected to a model by overlapping the frequency matrix and the colored matrix, resulting in an overlapped matrix.**

By overlapping the frequency matrix and the colored matrix, we derive an overlapped matrix which contains all

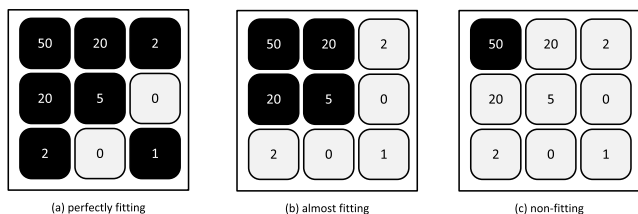
the information (i.e., the allowed behavior and observed behavior) from these two matrices. Note that the positions of the nine variants are fixed in all matrices, e.g., the bottom left cell corresponds to variant (0; 2+) in the frequency matrix, the colored matrix and the overlapped matrix. In Figure 5, the left matrix is a frequency matrix, which indicates the variants observed in the log, e.g., (0; 2+) is observed thirty-five times. The middle matrix is a colored matrix, which indicates that variants (0; 1) and (0; 2+) are allowed by the model. By overlapping them, we get the right matrix, i.e., an overlapped matrix. The overlapped matrix indicates that (0; 1) and (0; 2+) are observed fifty and thirty-five times, respectively, and they are allowed by the model. In contrast, (1; 0) and (2+; 2+) are observed ten and five times, respectively, and they are not allowed by the model. By summarizing the information in the overlapped matrix, we can claim that most observed behaviors are allowed by the model.

Through the variant matrix, we make a bridge to connect the log to the model, resulting in an overlapped matrix which indicates both allowed behavior and observed behavior. Next, we illustrate how to compute fitness and precision, based on the overlapped matrix.

**VI. FITNESS**

The first metric in the context of conformance is to quantify how much the real business process conforms to the specified behavior, i.e., how much the log fits the model. Based on the overlapped matrix defined in Section V-B, the proposed fitness is quantified as the extent to which the observed variants (in the OCEL) conform to the allowed variants (by the OCBC model).

Traditional process mining grants a perfect fitness (a value close to 1) to a model if it can replay all traces in the log from beginning to end. In comparison, a model has a poor fitness (a value close to 0) if it allows for little behavior seen in the event log. In this paper, we refer to the traditional process mining and propose solutions to compute fitness as a value between 0 and 1 in the context of OCELS and OCBC models.



**FIGURE 6.** Three overlapped matrices indicating different fitting situations.

As shown in Figure 6, three overlapped matrices indicate different fitting situations. More precisely, the first situation has a perfect fitness, since all observed behavior in the log is allowed by the model. In comparison, most observed behavior is allowed in the second overlapped matrix, describing an almost fitting situation. Differently, the third overlapped matrix describes a non-fitting situation, since

only one observed variant is allowed. The above discussion inspires that the fitness can be expressed as the ratio of the observed and allowed variants in all observed variants. Based on this idea, the fitness is 0 if none of the observed variants are allowed or 1 if all observed variants are allowed.

It is possible to take into consideration a threshold for the frequencies of variants to make the approach robust in terms of noise. More precisely, if the frequency of a variant is below the configured threshold, we consider that it is not observed in the log. The threshold can also be set as a ratio to consider the relative frequency when deciding if a variant is observed.

*Definition 14 (Fitness):* Let  $L$  be an OCEL,  $M$  be an OCBC model and  $P$  be a correlation pattern. Function  $fitnessP \in \mathcal{U}_L \times \mathcal{U}_{OCBCM} \times \mathcal{U}_P \rightarrow [0, 1]$  computes the fitness on the pattern level. We define the following notions:

- $fitnessP_1(L, M, P) = \frac{\{v \in V \mid freV(L, P, v) \geq 1\}}{\{v \in V_{CT} \mid freV(L, P, v) \geq 1\}}$ ,
- $fitnessP_2(L, M, P) = \frac{\{v \in V \mid freV(L, P, v) \geq \tau\}}{\{v \in V_{CT} \mid freV(L, P, v) \geq \tau\}}$  for some threshold  $\tau \in \mathbb{N}$ , and
- $fitnessP_3(L, M, P) = \frac{\{v \in V \mid freV_{\%}(L, P, v) \geq \tau\}}{\{v \in V_{CT} \mid freV_{\%}(L, P, v) \geq \tau\}}$  for some threshold  $\tau \in [0, 1]$ ,

where  $V = posV(M, P)$  is the set of allowed variants.

Four solutions are given in Definition 14 to calculate fitness on the pattern level (i.e., corresponding to some pattern  $P$ ).  $fitnessP_1$  only counts the number of variants which are allowed and observed, divided by the number of all observed variants. This solution is not applicable for real life logs, since it is sensitive to noise.  $fitnessP_2$  can deal with noise by setting a threshold  $\tau$  (i.e., an integer) to filter the infrequent variant. Similar to the second solution,  $fitnessP_3$  only changes the absolute frequency to a relative ratio.

**TABLE 1.** The fitness derived by different solutions for each overlapped matrix in Figure 6.

Solution	Threshold	fitness		
		Figure 6(a)	Figure 6(b)	Figure 6(c)
$fitnessP_1$	-	7/7(=1.0)	4/7(=0.57)	1/7(=0.14)
$fitnessP_2$	1	7/7(=1.0)	4/7(=0.57)	1/7(=0.14)
$fitnessP_2$	2	6/6(=1.0)	4/6(=0.67)	1/6(=0.17)
$fitnessP_2$	3	4/4(=1.0)	4/4(=1.0)	1/4(=0.25)
$fitnessP_2$	10	3/3(=1.0)	3/3(=1.0)	1/3(=0.33)
$fitnessP_2$	30	1/1(=1.0)	1/1(=1.0)	1/1(=1.0)
$fitnessP_3$	0.01	7/7(=1.0)	4/7(=0.57)	1/7(=0.14)
$fitnessP_3$	0.02	6/6(=1.0)	4/6(=0.67)	1/6(=0.17)
$fitnessP_3$	0.03	4/4(=1.0)	4/4(=1.0)	1/4(=0.25)
$fitnessP_3$	0.1	3/3(=1.0)	3/3(=1.0)	1/3(=0.33)
$fitnessP_3$	0.3	1/1(=1.0)	1/1(=1.0)	1/1(=1.0)

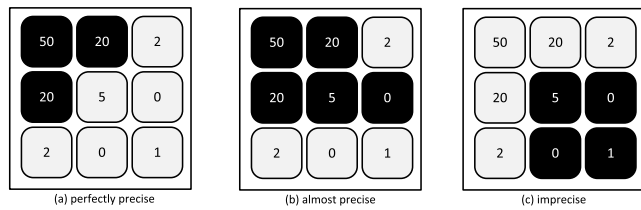
Next, we use the three overlapped matrices in Figure 6 to understand how to compute fitness with different solutions. In the first overlapped matrix (i.e., Figure 6(a)), seven variants are observed (seven cells with numbers greater than zero) and all these variants are allowed (cells colored in black). As a result, the fitness is 7/7 based on  $fitnessP_1$ . In the second

matrix, seven variants are observed, in which four of them are allowed and  $fitnessP_1$  returns 4/7. Following the same rule,  $fitnessP_1$  returns 1/7 for the third matrix, as shown in the first row in Table 1.

When the threshold is set as 1,  $fitnessP_2$  returns the same result. The number of observed variants (whose frequencies are above the threshold) decreases with increasing the threshold. For example, there are three observed variants when setting the threshold as 10, and all of them are allowed in the first and second matrix. As a result, the fitness is 3/3 based on  $fitnessP_2$ . In comparison, only one of the three observed variants is allowed in the third matrix, and  $fitnessP_2$  returns 1/3, as shown in the fifth row in Table 1. Based on a relative ratio,  $fitnessP_3$  computes fitness in a similar way to  $fitnessP_2$ .

**VII. PRECISION**

A good model needed to be as precisely as possible, i.e., if it does not allow for “too much” behavior. If a model allows for more behavior than necessary, it becomes too general and less informative as it no longer describes the actual process. Precision is another important criterion, which should be taken into consideration when quantifying the conformance. More precisely, precision evaluates how much behavior is allowed by the model which actually never happens in the log. It evaluates the conformance from another perspective and is different from fitness which evaluates whether the behavior in the log is possible with respect to the process model. A model having a poor precision is underfitting, i.e., it allows for behavior that is very different from what was seen in the event log.



**FIGURE 7.** Three overlapped matrices indicating situations with different precision.

As shown in Figure 7, three overlapped matrices indicate situations with different precision. The first situation has perfect precision, since all behavior allowed by the model is observed in the log. In comparison, since most allowed behavior is observed, the second overlapped matrix shows an almost precise situation. Differently, the third matrix describes an imprecise situation, since two of the four allowed variants are not observed and the other two are observed infrequently.

Based on discussing precision in different situations, this paper proposes solutions to compute precision in the context of OCEs and OCBC models. Intuitively, the precision can be quantified as the extent to which the allowed variants (by the OCBC model) are observed in the log based on the overlapped matrix defined in Section V-B. More precisely, the precision is calculated as the ratio of the observed and allowed

**TABLE 2.** The precision derived by different solutions for each overlapped matrix in Figure 7.

Solution	Threshold	precision		
		Figure 7(a)	Figure 7(b)	Figure 7(c)
$precisionP_1$	-	3/3(=1.0)	4/5(=0.8)	2/4(=0.5)
$precisionP_2$	1	3/3(=1.0)	4/5(=0.8)	2/4(=0.5)
$precisionP_2$	3	3/3(=1.0)	4/5(=0.8)	1/4(=0.25)
$precisionP_2$	10	3/3(=1.0)	3/5(=0.6)	0/4(=0.0)
$precisionP_2$	30	1/3(=0.33)	1/5(=0.2)	0/4(=0.0)
$precisionP_3$	0.01	3/3(=1.0)	4/5(=0.8)	2/4(=0.5)
$precisionP_3$	0.03	3/3(=1.0)	4/5(=0.8)	1/4(=0.25)
$precisionP_3$	0.1	3/3(=1.0)	3/5(=0.6)	0/4(=0.0)
$precisionP_3$	0.3	1/3(=0.33)	1/5(=0.2)	0/4(=0.0)
$precisionP_4$	-	0.91	0.71	0.33

variants in all allowed variants. For instance, the precision is 0 if none of the allowed variants are observed or 1 if all allowed variants are observed. Referring to the approach of computing fitness, we also take into consideration a threshold, i.e., we consider that a variant is not observed in the log if the frequency of the variant is below the configured threshold.

*Definition 15 (Precision):* Let  $L$  be an OCEL,  $M$  be an OCBC model and  $P$  be a correlation pattern. Function  $precisionP \in \mathcal{U}_L \times \mathcal{U}_{OCBCM} \times \mathcal{U}_P \rightarrow [0, 1]$  computes the precision on the pattern level. We define the following notions:

- $precisionP_1(L, M, P) = \frac{\{v \in V \mid freV(L, P, v) \geq 1\}}{|V|}$ ,
- $precisionP_2(L, M, P) = \frac{\{v \in V \mid freV(L, P, v) \geq \tau\}}{|V|}$  for some threshold  $\tau \in \mathbb{N}$ ,
- $precisionP_3(L, M, P) = \frac{\{v \in V \mid freV_{\%}(L, P, v) \geq \tau\}}{|V|}$  for some threshold  $\tau \in [0, 1]$ , and
- $precisionP_4(L, M, P) = \frac{\sum_{v \in V} -p_v \log_2(p_v)}{\log_2(|V|)}$  with  $p_v = \frac{freV(L, P, v)}{freV(L, P, V)}$ <sup>5</sup>,

where  $V = posV(M, P)$ .

Four solutions are given in Definition 15 to compute precision on the pattern level (i.e., corresponding to some pattern  $P$ ).  $precisionP_1$  only counts the number of variants which are allowed and observed, divided by the number of all allowed variants. The first solution is not applicable for real life logs since it is sensitive to noise. With the same filtering approach as used for computing fitness (cf. Definition 14),  $precisionP_2$  and  $precisionP_3$  deal with noise by setting a threshold  $\tau$ . Different from the previous solutions which use the frequency to compute precision,  $precisionP_4$  computes precision based on the extent to which the allowed variants are evenly observed in terms of frequency. More precisely, the precision is 0 if only one allowed variant is observed, and the precision is 1 if all allowed variants have the same frequency.

<sup>5</sup>This is based on the idea of entropy. As before, we assume that  $\frac{x}{0} = 1$ .



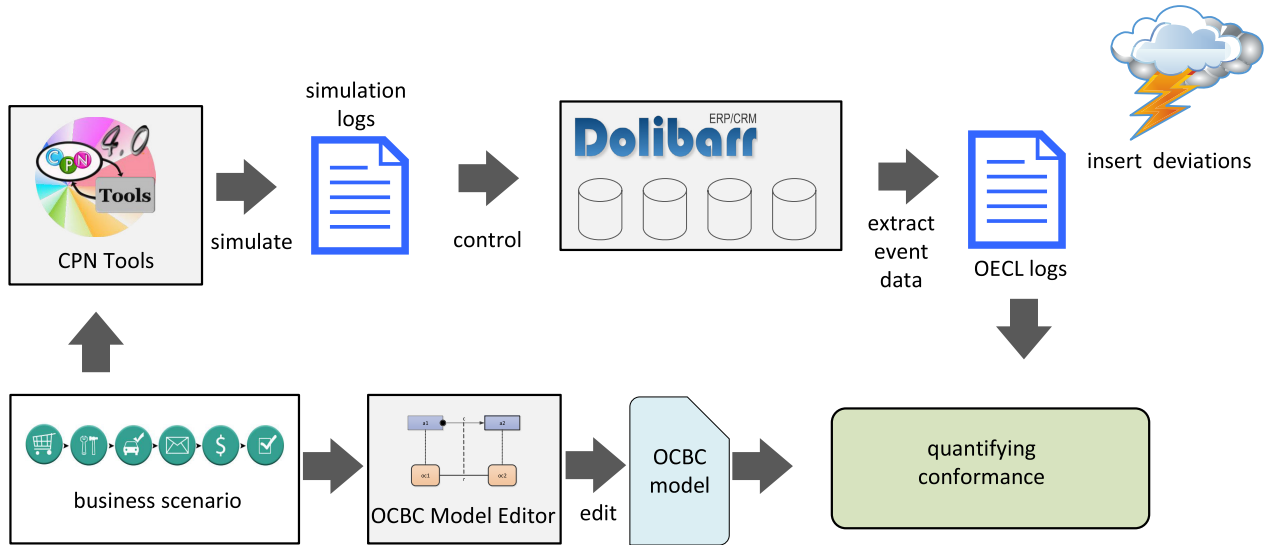


FIGURE 8. The approach to evaluate the OCBC conformance checking approach and tooling.

Next, we use the three overlapped matrices in Figure 7 to understand how to compute precision with different solutions. In the first overlapped matrix (i.e., Figure 7(a)), there are three allowed variants and all these variants are observed. As a result, the precision is 3/3 based on  $precisionP_1$ . In the second matrix, five variants are allowed, in which four are observed. As a result, the precision is 4/5 based on  $precisionP_1$ . Following the same rule,  $precisionP_1$  returns 2/4 for the third matrix, as shown in the first row in Table 2. When setting the threshold as 1,  $precisionP_2$  returns the same result. Note that, the number of observed variants decreases with increasing the threshold. For example, the number of observed variants in the third matrix drops to 1 with a threshold 3. As a result,  $precisionP_2$  returns 1/4 for the third matrix.  $precisionP_3$  computes precision in a similar way to  $precisionP_2$  based on a relative ratio. In comparison,  $precisionP_4$  computes precision based on entropy. The precision is 1 based on  $precisionP_4$  if the frequency of each allowed variant in Figure 7(a) is 30, indicating that allowed variants are equally observed. When changing the balance between allowed variants, the precision decreases. As shown in Figure 7(a), the precision is 0.91 when the frequency is 50, 20 and 20.<sup>6</sup>

## VIII. EVALUATION

In this section, the OCBC conformance quantifying approach is evaluated. Figure 8 shows details about the evaluation experimental environment.

More precisely, starting from a particular business scenario, the upper branch simulates the scenario to generate a log (representing observed behavior) and inject some known deviations into the log, the lower branch designs a model to describe the scenario (representing allowed behavior). At last, based on the generated log and reference model,

we verify if the proposed metrics can quantify these injected deviations accurately. Note that, the existing approaches are also implemented following the above process and compared with our approach.

### A. BUSINESS PROCESS

Based on a real ERP system named Dolibarr, we design an order-to-cash (OTC) business process with an informal notation in Figure 9 for the evaluation experiments. More precisely, the cardinality constraints between activities are indicated by the numbers on edges and the temporal order between activities is indicated by the arrows of edges. Consider the edge between the “create payment” activity and the “create invoice” activity as example. It requires that each “create invoice” event must be followed by corresponding “create payment” events, and the cardinalities on the edge indicates that each “create payment” event corresponds to one or more “create invoice” events and each “create invoice” event corresponds to precisely one “create payment” event [33].

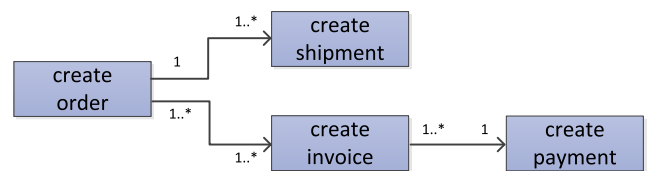


FIGURE 9. An informal model to describe the OTC business process.

Based on the designed OTC scenario, we use CPN (i.e., colored Petri net) Tools to create a simulation model involving multiple interacting entities [34], [35], as shown in Figure 8. A simulation log is created by running the OTC process in the simulation model with CPN Tools. Then the simulation log is interpreted to automatically operate the Dolibarr system with populating its database.<sup>7</sup>

$$6 - \frac{(50/90) \log_2(50/90) - (20/90) \log_2(20/90) - (20/90) \log_2(20/90)}{\log_2(3)} = 0.91.$$

<sup>7</sup>[http://www.win.tue.nl/ocbc/software/data\\_generation.html](http://www.win.tue.nl/ocbc/software/data_generation.html).

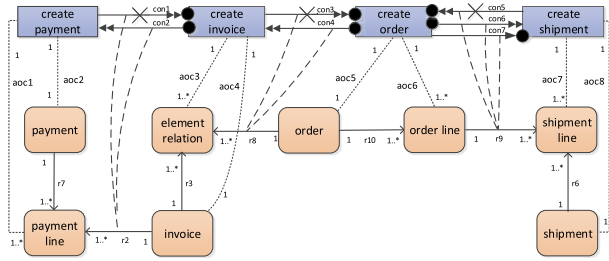


FIGURE 10. An OCBC model describing the order-to-cash process in ERP systems.

In Dolibarr system, the OTC process scenario involves eight tables, such as order, order line, shipment, shipment line, invoice, payment, payment line and element relation, as shown in Figure 12. There exist PK-FK (i.e., primary key and foreign key) relations between tables which indicate the references between table records. For instance, the order line  $o1$  refers to the order  $o1$ . By running the simulation, the tables of Dolibarr get filled with information about orders, invoices, shipments, etc.

Based on the OTC business process and the according data schema, we design an OCBC model, as shown in Figure 10. The class model is consistent with the data schema, i.e., eight classes correspond to eight tables and the class relations correspond to the PK-FK relations between tables. The seven behavioral constraints correspond to the restrictions assigned on the events in the OTC process. For instance, the constraint  $con2$  indicates each “create invoice” event must be followed by at least one corresponding “create payment” event.

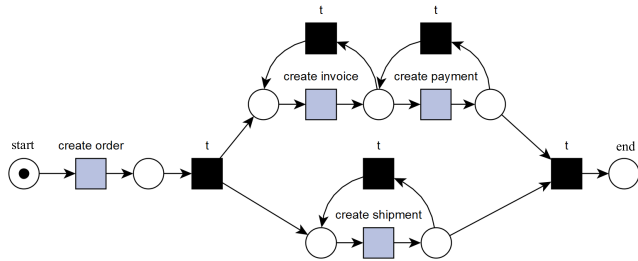


FIGURE 11. A Petri net to describe the OTC business process.

In order to conduct the comparison experiment, we also design a Petri net based on the OTC process, as shown in Figure 11. Note that, in order to enable the many-to-many relations between orders, invoices, payments and shipments, we have to add silent transitions in the net to create self-loop for “create invoice”, “create payment” and “create shipment”, due to the Petri net assumes a case notion. Figure 11 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.

### B. EVENT LOG

The conformance checking approach highly depends on the availability of event logs. After filling the Dolibarr system with information, we extract OCELS and XES event logs from the achieved tables as shown in Figure 12.

Table 3 and Table 4 briefly present the extracted OCEL [31], [32]. Table 3 represents the event records, where each row corresponds to a distinct event. Table 4 represents the relevant information of objects in the information systems. Figure 13 shows the extracted XES log with the method in [36] to relate events, resulting in a log with two cases  $o1$  and  $o2$ , i.e.,  $L = [ < create\ order, create\ shipment, create\ invoice, create\ shipment, create\ invoice, create\ payment >, < create\ order, create\ invoice, create\ shipment, create\ payment, create\ invoice, create\ payment > ]$ . Note that the XES log suffers convergence problems. More precisely, though  $ci2$  is performed only once in Dolibarr, it is contained by two cases as if it happens twice. Besides, the XES log also suffers divergence problems. More precisely,  $cp1$  and  $cp2$  (the two instances of “create payment”) cannot be distinguished in the case  $o2$  though they are performed on different documents in Dolibarr (i.e.,  $cp1$  is on  $ci2$  and  $cp2$  is on  $ci3$ ).

For controlled experiments, some deviations are added into the normal log to examine if these deviations can be quantified. In the normal logs, each “create invoice” event is followed by at least one “create payment” event, as shown in Figure 9. We insert deviations into the normal event logs to see if the inserted deviations can be identified. More precisely, we assume that the first “create payment” event does not occur, i.e., we remove  $p1$  from table “payment” and  $p11, p12$  from table “payment” and table “payment\_line” in Figure 12, respectively.

As a result,  $cp1$  is removed from the events in Table 3 and  $p1$  and  $p11$  are removed from objects in Table 4. Note that the relations related to the objects  $p1$  and  $p11$  are also missing. Figure 14 visualizes the OCEL after inserting the deviations. Accordingly, for the XES log, the event “create payment ( $cp1$ )” is removed from the cases  $o1$  and  $o2$ , as shown in Figure 15.

### C. QUANTIFYING THE CONFORMANCE

Like other modeling languages such as Petri nets, OCBC models support checking conformance on the behavioral perspective. In the experiment, we only focus on fitness to present the effectiveness of quantifying the conformance.

First, we check if the normal log has the perfect fitness, i.e., 1. In the normal scenario, each “create invoice” event is followed by at least one “create payment” event, indicated

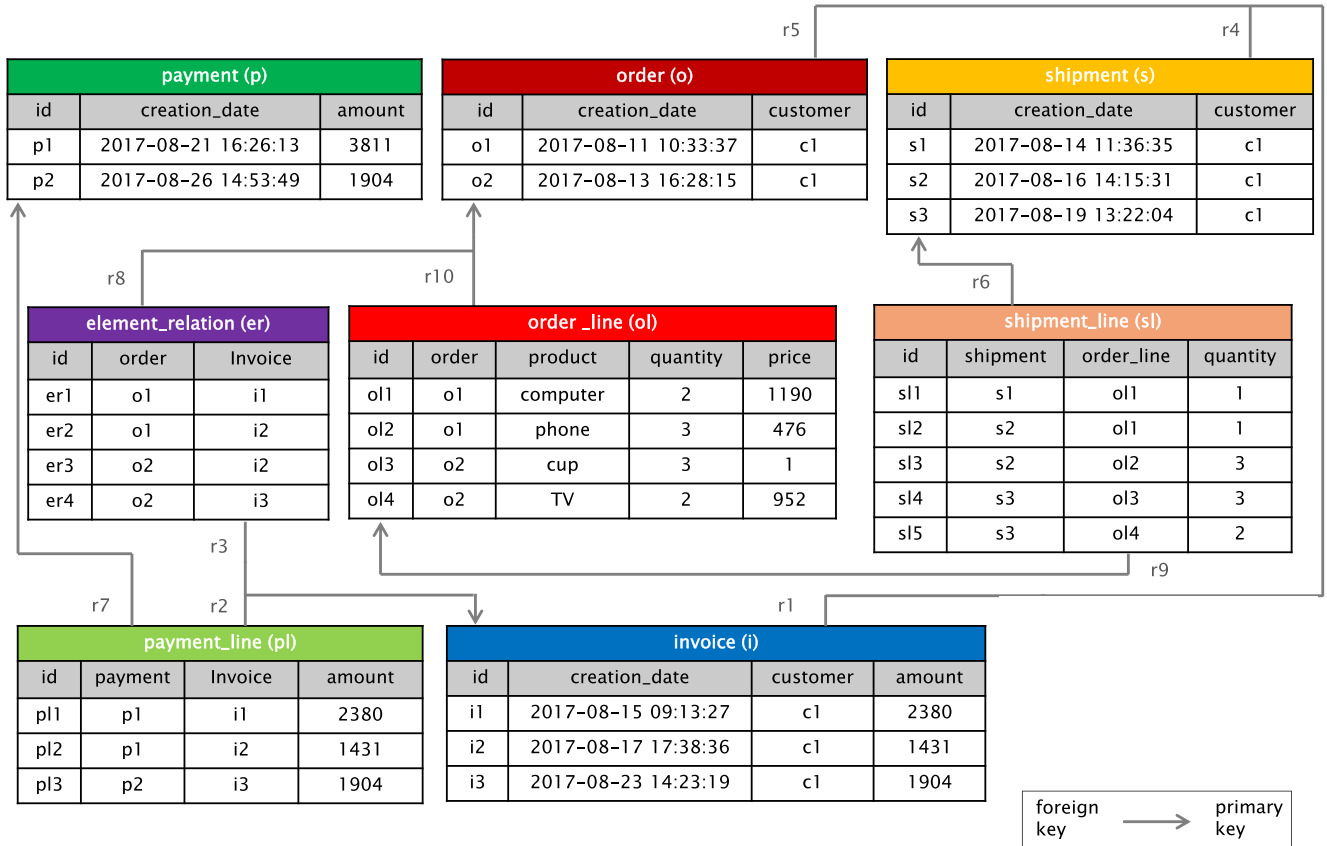


FIGURE 12. Tables and relations between them in the OTC scenario.

TABLE 3. 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
...	...	...	...	...	...	...	...	...	...	...
co1	create order	2017-8-11 10:33:37	{o1,o2}	{o1}	∅	∅	∅	∅	∅	∅
co2	create order	2017-8-13 16:28:15	{o3,o4}	{o2}	∅	∅	∅	∅	∅	∅
cs1	create shipment	2017-8-14 11:36:35	∅	∅	∅	∅	{s1}	{s1}	∅	∅
ci1	create invoice	2017-8-15 09:13:27	∅	∅	{er1}	{i1}	∅	∅	∅	∅
cs2	create shipment	2017-8-16 14:15:31	∅	∅	∅	∅	{s2}	{s2,s3}	∅	∅
ci2	create invoice	2017-8-17 17:38:36	∅	∅	{er2,er3}	{i2}	∅	∅	∅	∅
cs3	create shipment	2017-8-19 13:22:04	∅	∅	∅	∅	{s3}	{s4,s5}	∅	∅
cp1	create payment	2017-8-21 16:26:13	∅	∅	∅	∅	∅	∅	{p1,p2}	{p1}
ci3	create invoice	2017-8-23 14:23:19	∅	∅	{er4}	{i3}	∅	∅	∅	∅
cp2	create payment	2023-8-26 14:53:49	∅	∅	∅	∅	∅	∅	{p3}	{p2}
...	...	...	...	...	...	...	...	...	...	...

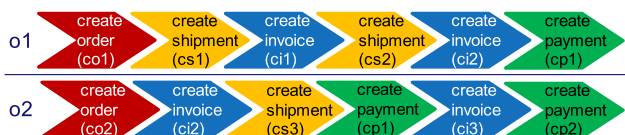


FIGURE 13. The generated XES log based the motivating data.

by the constraint *con2* in Figure 10. Next, we take *con2* as an example to show the process of quantifying the conformance.

In order to calculate the fitness, we correlate events into instances, and derive the overlapped matrix by overlapping the frequency matrix and the colored matrix. More precisely, we derive three instances after event correlation as shown in the left part in Figure 3. The left matrix is a variant matrix

and the middle left cell in the matrix corresponds to variant (0; 1). The right matrix is a frequency matrix and the number 3 in the middle left cell indicates that (0; 1) is observed three times in the log.

Figure 4 shows the process of mapping the constraint *con2* onto the variant matrix, resulting in a colored matrix, i.e., the right matrix with the middle left and bottom left cells colored in black, which indicates that variants (0; 1) and (0; 2+) are allowed by the model.

Figure 5 shows the process of overlapping the frequency matrix in Figure 3 and the colored matrix in Figure 4, resulting in an overlapped matrix. Based on the overlapped matrix, we calculate fitness with the first solution, which

TABLE 4. 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	payment		
...	...	...	...	...	...	...	...	...	...	...	...	...	...
ol1	order line		computer	∅	{o1}	∅	∅	∅	{s11}	∅	∅	∅	∅
ol2	order line		phone	∅	{o1}	∅	∅	∅	{s12}	∅	∅	∅	∅
ol3	order line		cup	∅	{o2}	∅	∅	∅	{s14}	∅	∅	∅	∅
ol4	order line		TV	∅	{o2}	∅	∅	∅	{s15}	∅	∅	∅	∅
o1	order	Mike		{ol1,ol2}	∅	{er1,er2}	∅	∅	∅	∅	∅	∅	∅
o2	order	Mike		{ol3,ol4}	∅	{er3,er4}	∅	∅	∅	∅	∅	∅	∅
er1	element relation			∅	{o1}	∅	∅	{i1}	∅	∅	∅	∅	∅
er2	element relation			∅	{o1}	∅	{i2}	∅	∅	∅	∅	∅	∅
er3	element relation			∅	{o2}	∅	{i2}	∅	∅	∅	∅	∅	∅
er4	element relation			∅	{o2}	∅	{i3}	∅	∅	∅	∅	∅	∅
i1	invoice	Mike		∅	∅	{er1}	∅	∅	∅	∅	{p11}	∅	∅
i2	invoice	Mike		∅	∅	{er2,er3}	∅	∅	∅	∅	{p12}	∅	∅
i3	invoice	Mike		∅	∅	{er4}	∅	∅	∅	∅	{p13}	∅	∅
s1	shipment	Mike		∅	∅	∅	∅	∅	{s11}	∅	∅	∅	∅
s2	shipment	Mike		∅	∅	∅	∅	∅	{s2,s13}	∅	∅	∅	∅
s3	shipment	Mike		∅	∅	∅	∅	∅	{s4,s15}	∅	∅	∅	∅
s11	shipment line		computer	{ol1}	∅	∅	∅	{s1}	∅	∅	∅	∅	∅
s12	shipment line		computer	{ol1}	∅	∅	∅	{s2}	∅	∅	∅	∅	∅
s13	shipment line		phone	{ol2}	∅	∅	∅	{s2}	∅	∅	∅	∅	∅
s14	shipment line		cup	{ol3}	∅	∅	∅	{s3}	∅	∅	∅	∅	∅
s15	shipment line		TV	{ol4}	∅	∅	∅	{s3}	∅	∅	∅	∅	∅
p11	payment line			∅	∅	∅	{i1}	∅	∅	∅	∅	{p1}	∅
p12	payment line			∅	∅	∅	{i2}	∅	∅	∅	∅	{p1}	∅
p13	payment line			∅	∅	∅	{i3}	∅	∅	∅	∅	{p2}	∅
p1	payment	Mike		∅	∅	∅	∅	∅	∅	∅	{p11,p12}	∅	∅
p2	payment	Mike		∅	∅	∅	∅	∅	∅	∅	{p13}	∅	∅
...	...	...	...	...	...	...	...	...	...	...	...	...	...

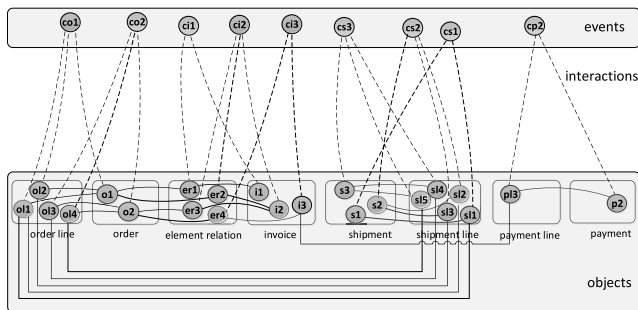


FIGURE 14. The OCEL after inserting the deviations.

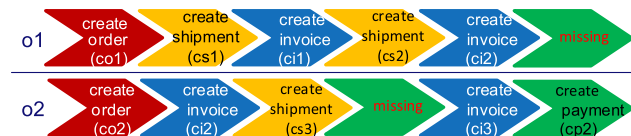


FIGURE 15. The generated XES log after inserting deviations.

counts the number of variants which are observed and allowed, divided by the number of all observed variants. In this way, the fitness is calculated as 1 since all observed variants are allowed.

Next, we calculate the fitness for the deviating log. As shown in Figure 19, only one has the payment event in the three derived instances after event correlation. Accordingly, in the frequency matrix, the number 2 in the top left cell indicates that (0; 0) is observed two times and the number 1 in the middle left cell indicates that (0; 1) is observed one time in the log. The left matrix is a variant matrix and the middle left cell in the matrix corresponds to variant (0; 1). The right

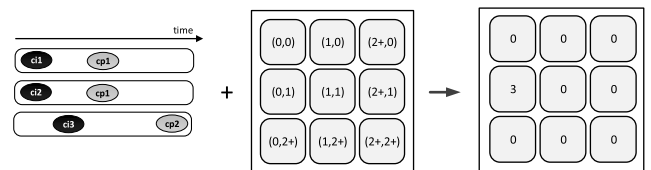


FIGURE 16. Mapping the normal log onto the variant matrix to show observed behavior.

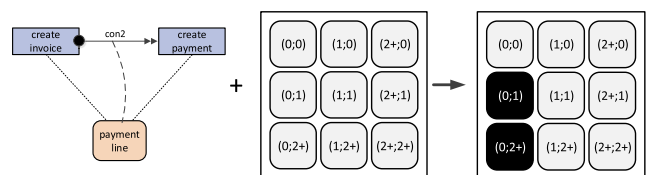


FIGURE 17. Mapping the constraint con2 onto the variant matrix to show allowed behavior.

matrix is a frequency matrix and the number 3 in the middle left cell indicates that (0; 1) is observed three times in the log.

The overlapped matrix in Figure 20 indicates that two variants (0; 1) and (0; 2+) are allowed by the model, and two variants (0; 1) and (0; 0) are observed in the log. With the first fitness solution, the fitness is 0.5, since only one variant is observed and allowed while two variants are observed.

#### D. COMPARISON

The conformance quantifying result derived by our object-centric approach shows that the inserted deviations can

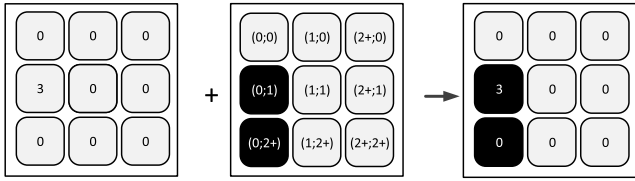


FIGURE 18. The normal log is connected to the constraint *con2* by overlapping the frequency matrix and the colored matrix, resulting in an overlapped matrix.

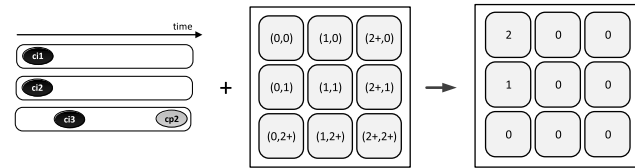


FIGURE 19. Mapping the deviating log onto the variant matrix to show observed behavior.

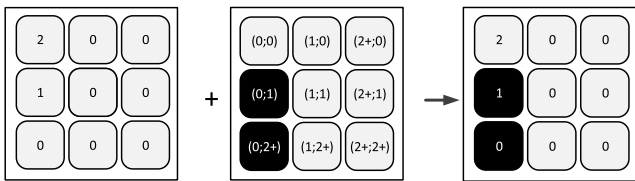


FIGURE 20. The deviating log is connected to the constraint *con2* by overlapping the frequency matrix and the colored matrix, resulting in an overlapped matrix.

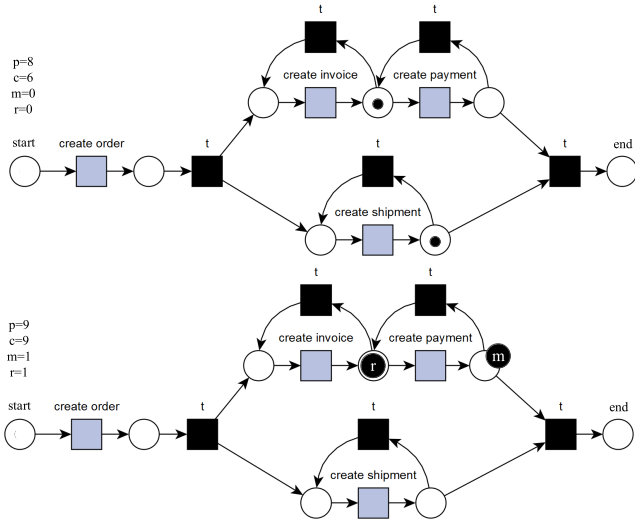


FIGURE 21. Replaying  $o1 = \langle co1, cs1, cf1, cs2, cf2 \rangle$ .

be detected. In this part, we apply the traditional techniques to the same data and compare its results with the result using our approach.

We first replay the normal log in Figure 13 on the Petri net in Figure 11. Initially, all places are empty and  $c = p = 0$ . Then the environment produces a token in “start” place. Therefore, the  $p$  counter is incremented and  $p = 1$ . Next we replay the case  $o1$  on the model and calculate the fitness. More precisely, we replay the first event  $co1$ , i.e., fire

transition “create order”. As a result, the  $c$  and  $p$  counters are incremented by 1 since “create order” consumes and produces one token, respectively. Therefore,  $c = 1$  and  $p = 2$  after firing transition “create order”. In order to replay the next events, we fire the silent transition “t” and  $c = 2$  and  $p = 4$  since “t” consumes one token and produces two tokens. Then we replay the second event  $cs1$ . Firing transition “create shipment” results in  $c = 3$  and  $p = 5$ . After replaying the third event (i.e.  $ci1$ )  $c = 4$  and  $p = 6$ . Similarly, after replaying the remaining events (i.e.,  $cs2$ ,  $ci2$  and  $cp1$ )  $c = 7$  and  $p = 9$ . After replaying the last event (i.e.  $cp1$ ), we fire the silent transition “t” and result in  $c = 9$  and  $p = 10$ . At the end, the environment consumes a token from “end” place. Hence the final result is  $c = p = 10$  and  $m = r = 0$ . Clearly, there are no problems when replaying the case  $o1$ , i.e., there are no missing or remaining tokens ( $m = r = 0$ ). The fitness is then calculated as  $fitness(\sigma, N) = \frac{1}{2}(1 - \frac{0}{10}) + \frac{1}{2}(1 - \frac{0}{10}) = 1$ . Following the above process, the fitness of the case  $o2$  is also 1.

Next, we replay the deviating log in Figure 15 on the Petri net in Figure 11. For the case  $o1$ , after replaying all the events (i.e.,  $co1$ ,  $cs1$ ,  $ci1$ ,  $cs2$  and  $ci2$ ),  $c = 6$  and  $p = 8$ . The tokens are in the place after “create invoice” and the place after “create shipment”, as shown in the top net in Figure 21. In order to finish the replaying, one token needs to be added into the place after “create payment” to trigger the silent transition “t”. After firing the transition “t”,  $c = 8$ ,  $p = 9$ ,  $m = 1$  and  $r = 1$ . At the end, the environment consumes a token from “end” place. Hence the final result is  $c = p = 9$  and  $m = r = 1$ , as shown in the bottom net in Figure 21. The fitness is then calculated as  $fitness(\sigma, N) = \frac{1}{2}(1 - \frac{1}{9}) + \frac{1}{2}(1 - \frac{1}{9}) = 8/9$ . For the case  $o2$ ,  $cp1$  is removed as a deviation. After replaying the third event  $cs3$ , the event  $ci3$  still can be triggered, since there is s self-loop for the transition “create invoice”. After replaying all events,  $m = r = 0$ , i.e., the case  $o2$  can be perfectly replayed on the model, and the fitness is 1.

In the deviating log, two invoices  $i1$  and  $i2$  are not paid. As we can see from the above example, the replay technique can only detect one deviating case, i.e.,  $o1$  in which  $i1$  has no corresponding payment event. It fails to efficiently detect deviations related to multiple instances, i.e.,  $o2$  in which  $i2$  is not paid. In comparison, our approach is more powerful in this situation. It can detect the two unpaid invoices  $i1$  and  $i2$  in a straight-forward manner.

### IX. CONCLUSION

In this paper, we proposed metrics to quantify the conformance between an OCEL log and an OCBC model in terms of fitness and precision. The quantifying task is split into two parts in this paper. The first part is connecting an OCEL log and an OCBC model by a variant matrix. The second task is to redefine three criteria, i.e., fitness and precision to quantify the conformance on the pattern level.

Our approach does not assume a global case notion which flattens the logs and models and leads to the

famous convergence and divergence problems. In this way, we overcome the problems of existing approaches that instances are considered in isolation. As a result, it is possible to quantify a range of conformance problems that would have remained undetected using traditional approaches.

## REFERENCES

- [1] 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, 2017.
- [2] 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, 2008.
- [3] 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 Manag.* Cham, Switzerland: Springer, 2011, pp. 345–361.
- [4] 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.
- [5] 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 Manag.* Cham, Switzerland: Springer, 2012, pp. 82–97.
- [6] 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.
- [7] A. Adriansyah, "Aligning observed and modeled behavior," Ph.D. thesis, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2014.
- [8] 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. Manag.*, vol. 13, no. 1, pp. 37–67, 2015.
- [9] W. M. P. van der Aalst, "Object-centric process mining: Dealing with divergence and convergence in event data," in *Proc. 17th International Conf.*, Oslo, Norway, Cham, Switzerland: Springer, Sep. 2019, pp. 3–25.
- [10] 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.
- [11] 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.
- [12] G. Li, "Process mining based on object-centric behavioral constraint (OCBC) models," Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2019.
- [13] B. Xiu and G. Li, "Diagnosing conformance between object-centric event logs and models," *IEEE Access*, vol. 11, pp. 110837–110849, 2023.
- [14] 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, 2008.
- [15] A. Rozinat, "Process mining: conformance and extension," Ph.D. thesis, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2010.
- [16] A. Berti and W. M. P. van der Aalst, "A novel token-based replay technique to speed up conformance checking and process enhancement," 2020, *arXiv:2007.14237*.
- [17] W. M. P. van der Aalst and A. Berti, "Discovering object-centric Petri nets," 2020, *arXiv:2010.02047*.
- [18] A. F. Ghahfarokhi, G. Park, A. Berti, and W. M. P. van der Aalst, "OCEL: A standard for object-centric event logs," *New Trends in Database and Information Systems* (Communications in Computer and Information Science), L. Bellatreche, M. Dumas, P. Karras, R. Matulevicius, A. Awad, M. Weidlich, M. Ivanovic, and O. Hartig, Eds., Cham, Switzerland: Springer, 2021, pp. 169–175.
- [19] J. C. Carrasquel, K. Mecheraoui, and I. A. Lomazova, "Checking conformance between colored Petri nets and event logs," in *Proc. Int. Joint Conf. Anal. Images Social Netw. Texts*, 2020, pp. 435–452.
- [20] J. Carrasquel and K. Mecheraoui, "Object-centric replay-based conformance checking: unveiling desire lines and local deviations," *Model. Anal. Inf. Syst.*, vol. 28, no. 2, pp. 146–168, 2021.
- [21] J. N. Adams and W. M. P. van der Aalst, "Precision and fitness in object-centric process mining," 2021, *arXiv:2110.05375*.
- [22] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. Enterprise Distrib. Object Comput. Conf. (EDOC)*, 2011, pp. 55–64.
- [23] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Towards robust conformance checking," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2010, pp. 122–133.
- [24] S. J. van Zelst and W. M. P. van der Aalst, "Online conformance checking: Relating event streams to process models using prefix-alignments," *Int. J. Data Sci. Anal.*, vol. 8, no. 3, pp. 269–284, 2019.
- [25] D. Schuster, S. J. van Zelst, and W. M. P. van der Aalst, "Conformance checking for trace fragments using infix and postfix alignments," *Cooperative Information Systems* (Lecture Notes in Computer Science), vol. 13591, Bozen-Bolzano, Italy, M. Sellami, P. Ceravolo, H. A. Reijers, W. Gaaloul, and H. Panetto, Eds., Cham, Switzerland: Springer, 2022, pp. 299–310.
- [26] M. Estañol and J. Munoz-Gama, "Conformance checking in UML artifact-centric business process models," *Softw. Syst. Model.*, vol. 18, no. 3, pp. 2531–2555, 2019.
- [27] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2010, pp. 211–226.
- [28] J. Munoz-Gama and J. Carmona, "Enhancing precision in process conformance: Stability, confidence and severity," in *Proc. Comput. Intell. Data Mining (CIDM) Symp.*, 2011, pp. 184–191.
- [29] 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 Manag.* Cham, Switzerland: Springer, 2012, pp. 137–149.
- [30] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, 2012.
- [31] G. Li, Renata M. 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)*. IEEE, pp. 203–210x.
- [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] B. Xiu, G. Li, and Y. Li, "Discovery of object-centric behavioral constraint models with noise," *IEEE Access*, vol. 10, pp. 88769–88786, 2022.
- [34] W. M. P. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach*. Cambridge, MA, USA: MIT Press, 2011.
- [35] K. Jensen, "Coloured Petri nets," in *Advances in Petri Nets 1986, Part I on Petri Nets: Central Models and Their Properties*. Berlin, Germany: Springer-Verlag, 1987, pp. 248–299.
- [36] W. van der Aalst, *Process Mining: Data Science in Action*, Berlin, Germany: Springer-Verlag, 2016.



**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 Harbin University of Technology, in 2012, and the M.Sc. degree in management science and engineering and the Ph.D. degree in information system (IS) from the 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.

...