Controlling Break-The-Glass Through Alignment

Arya Adriansyah, Boudewijn F. van Dongen, Nicola Zannone
Eindhoven University of Technology
{a.adriansyah,b.f.v.dongen,n.zannone}@tue.nl

***Abstract*—Modern IT systems have to deal with unpredictable situations and exceptions more and more often. In contrast, security mechanisms are usually very rigid. Functionality like break-the-glass is thus employed to allow users to bypass security mechanisms in case of emergencies. However, break-the-glass introduces a weak point in the system. In this paper, we present a flexible framework for controlling the use of break-the-glass using the notion of alignments. The framework measures to what extent a process execution diverges from the specification (i.e., using optimal alignments) and revokes the exceptional permissions granted to cope with the emergency when the severity of deviations cannot be tolerated. For the quantification of the severity of deviations, we extend alignment-based deviation analysis techniques by supporting the detection of high-level deviations such as activity replacements and swaps, hence providing a more accurate diagnosis of deviations than classical optimal alignments.**

I. Introduction

The basic notion of enforcement relies on the idea that infringements (i.e., deviations from policies and procedures) are violations and as such should not be permitted [1], [2]. This approach, however, is too inflexible to be used in dynamic and open environments like healthcare. While posing strict constraints on the access to sensitive information, the system has also to cope with the potential exceptions raised in case of emergencies.

The inflexibility of existing enforcement mechanisms often forces users to bypass them and just switch off security measures, which leads to insecurity. Most healthcare systems, for instance, include "break-the-glass" functionality which allow users to bypass access control rules in emergency situations [3], [4]. The deployment of break-the-glass functionality, however, introduces a weak point in the system that can be misused [5]. To regulate the use of break-the-glass and thus reduce security risks, it is advisable to develop flexible yet automated tools able to analyze user behavior at run-time and take compensation actions when the consequences of infringements cannot be tolerated.

Several approaches [6], [7] have been proposed to check compliance of user behavior with specification. Although these techniques are able to detect that a deviation occurred, they do not explicitly identify its root causes, making it difficult to quantify its severity. In contrast, the notion of alignments [8], [9] provides a robust approach to determine the root causes of deviations. However, constructing optimal alignments is computationally expensive. Existing techniques [10] impose restrictions on the type of optimal alignments to compute them efficiently. In particular, alignments only identify low level deviations, i.e. insertions and suppressions. Nonetheless, the severity of deviations should be assessed in terms of high level deviations like replacements and swaps [11]. Therefore, low level deviations have to be analyzed and possibly related for a diagnosis of the actual deviations which occurred. Identifying low level deviations and then using them to diagnose high level deviations, however, may lead to misinterpret the root causes of deviations. The fundamental problem is that, although high level deviations can be seen as combinations of low level deviations, their severity cannot be defined in terms of those deviations [12].

In this paper we propose a flexible framework for controlled break-the-glass based on the notion of alignments. In case of an emergency, users may require exceptional permissions. The framework grants users such permissions and thus allows them to deviate from the specification in order to face the emergency. However, deviations can only be within a certain range: if the severity of those deviations becomes too high, the framework revokes the exceptional permissions.

We use online and offline conformance checking based on alignments to assess the severity of deviations. Online conformance checking measures the extent user behavior deviates from specification at run-time. Thus, it does not penalize executions that did not reach proper termination according to the model. This, however, may result in an underestimation of deviations. Offline conformance checking is performed after process execution is declared to be complete. Thus, it considers improper termination as deviations.

To provide a more accurate diagnosis of user behavior and deviations, we extend alignment techniques to support the detection of replacements and swaps. In particular, we propose constructs that explicitly represent these types of deviations in the process model. The technique has been implemented as a ProM plug-in and evaluated using both synthetic and real-life logs.

The paper is structured as follows. Section II introduces preliminary concepts. Section III presents our approach for controlled break-the-glass, and Section IV presents an extension of alignment techniques to deal with replacements and swaps. Section V presents experiment results. Finally, Section VI concludes the paper.

II. Preliminaries

Process models describe how processes should be carried out. We consider process models in the form of Petri nets. A Petri net consists of transitions, places, arcs connecting places and transitions, and a labeling function. The labeling function maps transitions to the activities they represent, while places and arcs describe the routing of activities and dependency between them.

Fig. 1 shows a simplified procedure of a healthcare treatment in a hospital. A process instance starts when a patient

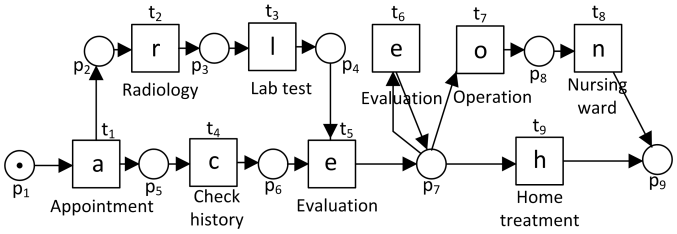


Figure 1. Example of treatment process

$$\sigma_1 = \langle \mathsf{a}, \mathsf{r}, \mathsf{l}, \mathsf{c}, \mathsf{e}, \mathsf{e}, \mathsf{o}, \mathsf{n} \rangle \quad \sigma_2 = \langle \mathsf{a}, \mathsf{l}, \mathsf{r}, \mathsf{c}, \mathsf{e} \rangle \quad \sigma_3 = \langle \mathsf{a}, \mathsf{r}, \mathsf{l}, \mathsf{e}, \mathsf{l}, \mathsf{t} \rangle$$

Figure 2. Example of traces for the model in Fig. 1

makes an Appointment. Then, the patient needs to go through Radiology check, followed by Lab test. Meanwhile, a Check history on his/her previous medical record is performed. Gathered information is then Evaluated and decision is made whether the patient should go on Operation or Home treatment instead. Evaluation can be performed multiple times, but it has to be performed at least once. After an operation, a patient stays at Nursing ward until his/her condition permits dehospitalization. Notice that transitions t_5 and t_6 are both labeled with activity Evaluation in Fig. 1.

The state of a Petri net is represented by a *marking*, i.e. a multiset of tokens on the places of the net. A Petri net has an initial marking and a final marking. The initial marking of the net in Fig. 1 consists of one token in place p_1. This place is the input place of transition Appointment, therefore only transition Appointment is enabled according to the net and marking. When a transition is executed (i.e., *fired*), a token is taken from each of its input places and a token is added to each of its output places. A Petri net terminates properly if it reaches its final marking (i.e., one token in place p_9 in Fig. 1). A sequence of transitions from the initial to the final marking of a Petri net is a *complete run* of the net.

A *trace* is a sequence of activities, representing a process instance. An instance of an activity in a trace is called an *event*. An *event log* is a multiset of traces. Fig. 2 shows some examples of traces for the process in Fig. 1, where activities are abbreviated according to their initial letter. Note that an activity may occur many times in a trace, e.g. Evaluation can be performed multiple times (see trace σ_1 in Fig. 2).

Not all traces can be reproduced by the net, i.e. not all traces perfectly fit the process description. If a trace perfectly fits a Petri net, each "move" in the trace, i.e. an activity observed in the trace, can be mimicked by a "move" in the model, i.e. a transition fired in the net. Moreover, after all activities in the trace are mimicked, the net reaches its final marking. σ_1 in Fig. 2 is an example of a perfectly fitting trace for the net in Fig. 1.

In cases where deviations occur, some movements that occur in the trace cannot be mimicked by the net or vice versa. We explicitly denote "no move" by $\gg$. A *legal movement* is a pair (x, y) such that

- (x, y) is a *synchronous move* if x is an activity in the trace and y is a transition in the net, and
- (x, y) is a *move on log* if x is an activity in the trace and y is $\gg$,

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|} \hline \mathsf{a} & \mathsf{l} & \mathsf{r} & \mathsf{c} & \gg & \mathsf{e} & \gg \\ \hline \mathsf{a} & & \mathsf{r} & \mathsf{c} & \mathsf{l} & \mathsf{e} & \mathsf{h} \\ t_1 & \gg & t_2 & t_4 & t_3 & t_5 & t_9 \\ \hline \end{array} \quad \gamma_2 = \begin{array}{|c|c|c|c|c|c|} \hline \mathsf{a} & \mathsf{l} & \mathsf{r} & \mathsf{c} & \gg & \mathsf{e} \\ \hline \mathsf{a} & & \mathsf{r} & \mathsf{c} & \mathsf{l} & \mathsf{e} \\ t_1 & \gg & t_2 & t_4 & t_3 & t_5 \\ \hline \end{array}$$

Figure 3. An alignment (left) and a prefix alignment (right) between trace σ_2 in Fig. 2 and the model in Fig. 1

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline \mathsf{a} & \mathsf{l} & \mathsf{r} & \mathsf{c} & \gg & \mathsf{e} & \gg & \gg \\ \hline \mathsf{a} & & \mathsf{r} & \mathsf{c} & \mathsf{l} & \mathsf{e} & \mathsf{o} & \mathsf{n} \\ t_1 & \gg & t_2 & t_4 & t_3 & t_5 & t_7 & t_8 \\ \hline \end{array}$$

Figure 4. A non optimal alignment between trace σ_2 in Fig. 2 and the model in Fig. 1

- (x, y) is a *move on model* if x is $\gg$ and y is a transition in the net.

A *(prefix) alignment* between the trace and the net is a sequence of legal movements such that its sequence of activities (ignoring $\gg$) yields the original trace, and its sequence of transitions (ignoring $\gg$) yields a (prefix of a) complete run of the net. Take for example the Petri net in Fig. 1 and trace σ_2 in Fig. 2. γ_1 in Fig. 3 is an alignment between σ_2 and the net. The top row of γ_1 shows the sequence of "moves" in the trace, and the bottom row shows the sequence of "moves" in the model (both ignoring $\gg$). For every transition, we indicate the corresponding activity on top of it.

Given a trace representing an incomplete process execution, an alignment between the trace and the model highlights the transitions that still need to be fired to reach proper termination as deviations. In contrast, prefix alignments highlight deviations without penalizing for non-completion. For example, γ_2 in Fig. 3 is a prefix alignment between the net in Fig. 1 and trace σ_2 in Fig. 2.

For a given process model and a trace, there can be more than one possible (prefix) alignment. For instance, γ_3 in Fig. 4 is another possible alignment between trace σ_2 and the net in Fig. 1. The quality of a (prefix) alignment is measured based on a predefined *cost function* that defines the cost of movements in alignments. An *optimal (prefix) alignment* between a process model and a trace is the one that has the least total cost according to the cost function. Various cost functions can be defined, depending on the application domain and purpose of the analysis. For example, the standard cost function [9] defines the cost of each move on model/move on log equal to 1 and the cost of each synchronous move between an activity and a transition labeled with the same activity equal to 0, and the cost is $+\infty$ otherwise. The total cost of alignment γ_1 is 3 and the total cost of alignment γ_3 is 4. Thus, γ_1 is better than γ_3 according to the standard cost function. Since there is no other alignment between the trace and the model that has total cost lower than 3, γ_1 is an optimal alignment.

Constructing optimal (prefix) alignments requires exploration of the (possibly infinite) state space of a Petri net, which is computationally expensive. However, existing approaches (e.g., [8], [10]) exploit properties of both Petri nets and traces to construct optimal alignments efficiently.

III. Controlled Break-The-Glass

In this section, we present a flexible enforcement mechanism that enables a controlled use of break-the-glass functionality. Its architecture is given in Fig. 5. In the remainder of

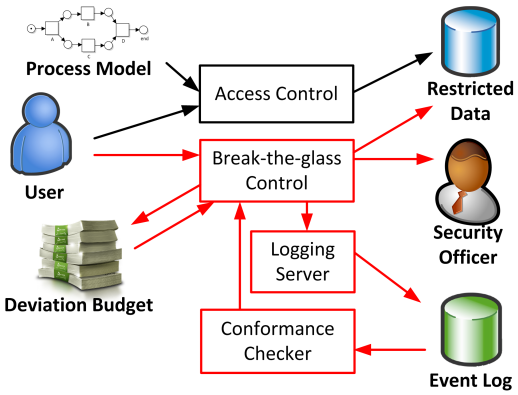


Figure 5. Controlled Break-the-glass Architecture

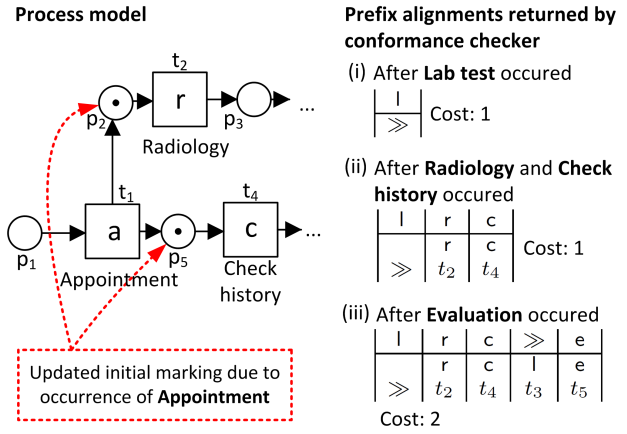


Figure 6. Updated initial marking of the process model, and prefix optimal alignments returned by conformance checker for the running example.

the section, we introduce the basic concepts and provide an overview of the architecture.

In case of an emergency, users may need to take actions that deviate from the specification to react to the emergency. A user may execute an activity which is not allowed by the specification (*insertion*). For instance, a doctor may perform additional tests to better evaluate the patient's health condition. A user may not execute an activity which is required by the specification (*suppression*). A user may execute an activity instead of another activity (*replacement*). For instance, a doctor may perform a CT-scan instead of an MRI test because of the temporary unavailability of technical equipment. A user may execute activities in a different order than the one defined in the specification (*swap*). Each deviation is associated with a *cost* that represents the severity of the deviation.

The *process model* defines the allowed behavior of the system. The *access control* mechanism keeps track of the state of the process execution and blocks the execution of those activities that are not compliant with the process model. At any point of the process execution, users can invoke the *break-the-glass control* to perform actions that they are usually not authorized to perform. This functionality takes priority over the access control mechanism allowing the user to perform the requested actions. While deviations from the specification may be necessary to react to the emergency, they pose security risks (e.g., data misuse). Therefore, the use of the break-the-glass functionality needs to be monitored. To this end, the break-the-glass control activates the *logging server* to record the user behavior in an *event log* and, therefore, make users accountable for their actions.

When a user invokes break-the-glass functionality, a *deviation budget* is assigned to such an invocation. Intuitively, the deviation budget represents to what extent users can deviate from the specification. The amount of the assigned budget may depend on the business value of the process or on the user requesting the exceptional permissions. Every time a user deviates from the specification, the cost of the deviation is subtracted from the deviation budget. When the deviation budget is overspent, the exceptional permissions are revoked, and the security officer is notified about the infringement.

The diagnostic information necessary to assess the severity of deviations is obtained through the *conformance checker*. This component takes as input a process model and an event log, and determines the *optimal alignment* between them by replaying the event log over the process model. The state of the process execution in which break-the-glass is invoked, is used to determine the initial marking on the process model. The conformance checker determines the cost-optimal alignment starting from this marking using the defined cost function.

The severity of deviation is assessed both *online* and *offline*. Online conformance checking is performed at run-time by replaying the trace of executed activities over the process model. Since traces that have not reached proper termination according to the model should not be penalized, prefix alignments are used to provide diagnostics information about deviations. However, online conformance checking may underestimate the total cost of deviations if the process execution ends without reaching proper termination. Offline conformance checking, which is performed after the process execution is declared to be complete, addresses this issue by also penalizing activities that should have been performed according to the model, but do not occur in the trace. Both online and offline conformance checking provide diagnostics information about the deviations beyond simple severity measurements.

Take for example the process in Fig. 1. Suppose that a user invokes break-the-glass after Appointment and a deviation budget equal to 1 is assigned to the user. The break-the-glass control activates the logging server which logs the process instance; conformance checking is performed online using the standard cost function. The initial state used for conformance checking is the marking of the net after firing t_1 (Appointment) as shown in Fig. 6 (left). After the user invoked break-the-glass, he performs Lab test. Fig. 6(i) shows the optimal prefix alignment between the recorded trace and the model, returned by the checker. As shown in the figure, the checker recognizes the event as a move on log.

After Lab test, the user performs activities Radiology, Check history, and Evaluation consecutively. The first two events are allowed according to the model, and the conformance checker returns the prefix alignment in Fig. 6(ii). However, after the occurrence of Evaluation, conformance checking returns an optimal prefix alignment with deviation cost of 2 (Fig. 6(iii)). The allocated deviation budget is overspent. Thus, the exceptional permissions are revoked and the security officer is informed about the infringement.

$\gamma_4 =$

a	r	l	≫	e	l	t
a	r	l	c	e		h
t_1	t_2	t_3	t_4	t_5	≫	t_9

Figure 7. An alignment between trace σ_3 in Fig. 2 and the model in Fig. 1, showing replacement of activity h with activity t

Suppose that the deviation budget assigned to the user is 2 (instead of 1). In this case, the user would still be allowed to proceed with the execution of the process. If the instance is declared to be complete after Evaluation, conformance checker then computes offline an optimal alignment between trace $\langle l, r, c, e \rangle$ and the process model with initial marking in Fig. 6. The optimal alignment has cost of 3 (a move on model for t_9(Home treatment) is added to the alignment in Fig. 6(iii)). Since the allocated deviation budget is overspent, an alert is sent to the security officer.

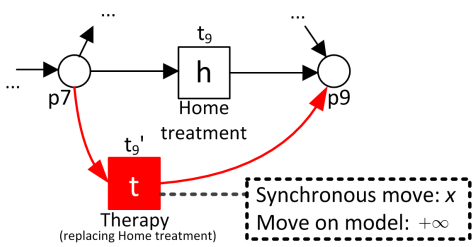
Existing alignment techniques only allow moves on log, moves on model, and synchronous moves between an activity and a transition labeled with the same activity. Therefore, alignments obtained using such techniques only show deviations of types *insertion* (i.e., moves on log) and *suppression* (i.e., moves on model). *Replacements* and *swaps* have to be identified as combinations of such deviations. For instance, the move on log and move on model for Lab test in Fig. 6 correspond to a swap between Lab test and Radiology. However, analyzing replaced and swapped activities from identified insertions and suppressions may lead to undesirable results [12]. In our example, if the cost of the swap is 1, the total cost of deviations after execution of Evaluation is still within deviation budget and so exceptional permissions should not have been revoked from the user. Therefore, it is desirable to identify replacements and swaps directly in alignments. In the next section, we show how to extend existing approaches to deal with replacements and swaps.

IV. Identifying Deviations

Given a (prefix) alignment, *replacements* manifest in the alignment as synchronous moves between activities and transitions that do not refer to the same activities. Take for example the model in Fig. 1 and trace σ_3 in Fig. 2. Suppose that a patient may undergo a therapy instead of home treatment in cases of emergency, but doing it in a non-emergency case is not recommended. Thus, doing Therapy instead of Home treatment is considered a replacement deviation with low cost (Notice that activity Therapy is not modeled in the model of Fig. 1). To consider this in the deviation analysis, let consider cost function δ_1 where the cost of all movements are the same as the standard cost function, except for the cost of the synchronous move between activity Therapy and transition t_9 (Home treatment) that is set to 1.

With cost function δ_1, the expected optimal alignment between σ_3 and the model is represented by alignment γ_4 in Fig. 7. Here, the synchronous move between activity Therapy and transition t_9 (Home treatment) shows the *replacement* of activity Home treatment with activity Therapy.

To efficiently compute optimal alignments, existing approaches (e.g., [8], [10]) cannot construct optimal alignments that show replacements such as γ_3 because they assume that synchronous moves can only occur between an activity and a transition labeled with the same activity. We overcome this limitation by explicitly modeling replacements in the net. For all pairs of activities $a, a', a \neq a'$ where a can be replaced by a', we duplicate all transitions labeled with a and map the duplicates to activity a' instead of a. We call the duplicates *substitution transitions*. Fig. 8 shows a substitution transition which models the replacement of activity Home treatment with activity Therapy. As shown in the figure, firing substitution transition t'_9 leads to the same state of the net as if transition t_9 is fired.



Cost of replacing **Home treatment** with **Therapy**: *x*

Figure 8. Modeling a replacement of activity Home treatment with activity Therapy as transition t'_9

$\gamma_5 =$

a	r	l	≫	e	l	t
a	r	l	c	e		t (replacing h)
t_1	t_2	t_3	t_4	t_5	≫	t'_9

Figure 9. An optimal alignment between trace σ_3 in Fig. 2 and the net of Fig. 1 extended with substitution transitions

We extend the cost function for insertions and suppressions with the cost of movements involving substitution transitions. The cost of synchronous moves (x, y) where x is an activity and y is its substitution transition, is the cost of the replacement defined by the substitution transition. The cost of move on model for substitution transitions is infinitely high (i.e., $+\infty$). This way, there cannot be any move on model for substitution transitions in any (prefix) alignment. An optimal alignment between trace σ_3 in Fig. 2 and the net of Fig. 1 augmented with the substitution transition in Fig. 8, using the extended cost function derived from δ_1, is shown in Fig. 9.

Translating the deviations in γ_5 (Fig. 9) back to deviations in the original model is done by considering synchronous moves that involve substitution transitions as replacements. Alignment γ_5 shows a synchronous move between substitution transition t'_9 and activity Therapy. Therefore, we know that execution of activity Home treatment has been replaced by activity Therapy. This is exactly the same interpretation that we get from the optimal alignment γ_4 in Fig. 7.

Swapped activities can be identified with a similar approach. Consider again the model in Fig. 1, trace σ_2 in Fig. 2, and their alignment γ_1 in Fig. 3. Suppose that activity Radiology can be swapped with activity Lab test without any cost. Instead of having one insertion (i.e., activity Lab test) and two suppressions (i.e., activities Lab test and Home treatment) as shown by γ_1, the deviations should be a pair of swapped activities (i.e., Radiology with Lab test) and one suppression (i.e., Home treatment).

We translate the problem of finding swapped activities into the problem of finding a (prefix) alignment between them.

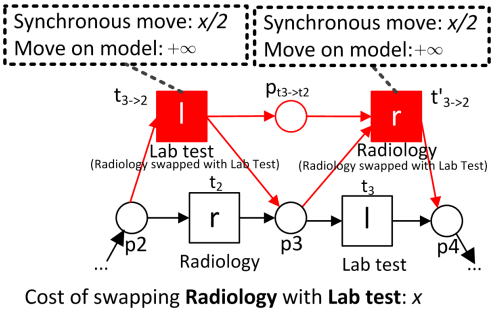


Figure 10. A pair of *interchange transitions* representing activity Radiology swapped with Lab test

$$\gamma_6 = \begin{array}{|c|c|c|c|c|c|} \hline a & l & r & c & e & \gg \\ \hline a & l(\text{r swapped with l}) & r(\text{r swapped with l}) & c & e & h \\ t_1 & t_{3\to 2} & t'_{3\to 2} & t_4 & t_5 & t_9 \\ \hline \end{array}$$

Figure 11. An optimal alignment between trace σ_2 in Fig. 2 and the net of Fig. 1 extended with the interchange transitions in Fig. 10

We augment the original model with *interchange transitions* that model swapped activities explicitly. Let a and a' be two activities where a can be swapped with a'. For all pairs of transitions (t_1, t_2) where t_1 and t_2 are labeled a and a' respectively, we duplicate t_1 and t_2 as a pair of *interchange transitions* $t_{2\to 1}$ and $t'_{2\to 1}$ respectively and reverse their label. Moreover, a place $p_{t_2\to t_1}$ is added to connect $t_{2\to 1}$ and $t'_{2\to 1}$ such that proper completion on the model is reached if and only if either both transitions are fired or none of them are fired. Fig. 10 shows an example of interchange transitions that explicitly models swapping activity Radiology with Lab test (i.e., swapping t_2 with t_3). Every firing of interchange transition $t_{3\to 2}$ must be followed by firing of $t'_{3\to 2}$, otherwise there is a remaining token in place $p_{t_3\to t_2}$ which implies that proper termination is not reached.

The cost of swapping two activities is distributed equally among the corresponding interchange transitions. In addition, we set the cost of moves on model for interchange transitions to $+\infty$. This way, moves on model on interchange transitions cannot occur in any optimal alignment between the net and traces. Fig. 11 shows an optimal alignment between σ_2 and the net augmented with the interchange transitions in Fig. 10 with respect to this modified cost function.

Swaps can be identified by pairing synchronous moves that involve interchange transitions. The optimal alignment γ_6 in Fig. 11 shows two synchronous moves that involve interchange transitions $t_{3\to 2}$ and $t'_{3\to 2}$. By pairing them and then refer to their labels, we know that activity Radiology has been swapped with activity Lab test.

V. Experiments

We have implemented the proposed conformance checking technique as a ProM 6 plug-in, publicly available at www.processmining.org. We performed experiments to evaluate the performance of the technique and obtain insight into the definition of the cost function. We refer to the full paper [13] for a more complete discussion on these issues.

The event log used in the experiments was obtained by generating a number of traces from a process model. These traces were then modified to introduce deviations of different types (first block in Table I). Three groups of experiments were performed. For each experiment, we report the identified deviations, the total cost of the optimal alignment (TC), and the average calculation time per trace.

In the first group of experiments (**Exp. 1**), we use existing alignment techniques (e.g., [8]) to detect deviations which occurred. These techniques consider only moves on log (INS) and moves on model (SUP) when determining deviations. In the experiments, we assigned a cost to every move on log and move on model equal to 1. The total cost of the optimal alignment is therefore equal to the number of identified deviations in the trace. The results of this group of experiments are presented in the second block of Table I. The results show that, although existing techniques are efficient, they overestimate the cost of alignments as replacements and swaps are detected as combinations of moves on log and moves on model. As a consequence, the low level alignment cannot be used to construct high level deviations. For instance, in trace 6 we expect two replacements (first block of Table I). At a low level, they would correspond to four deviations since replacements can be identified as a combination of a move on log and a move on model. In contrast, the optimal alignment for trace 6 has total cost of 3, given by an insertion and two suppressions. This example shows that if the cost function does not include the cost of replacements and swaps, these deviations may not be diagnosed from the alignment.

The aim of the second and third groups of experiments is to evaluate the ability of the method proposed in this work to identify deviations without any knowledge in advance about occurring deviations. In the second group of experiments (**Exp. 2**), we did not consider any prior knowledge about deviations. Thus, we use a standard cost function where the cost of all possible deviations (including replacements and swaps between all pairs of activities) is equal to 1. The results are presented in the third block of Table I. In general, the results of the second group coincide with the first block in the table. An exception is given by trace 11, for which the plug-in found a different interpretation for the deviations which occurred (i.e., three insertions and two replacements instead of four insertions, one suppression and one swap). However, a closer look at the trace revealed that the alignment determined by the plug-in does not correspond to a valid interpretation of the deviations which occurred. The alignment shows the replacement of two pairs of incompatible activities. This highlights the importance of assigning appropriate costs to deviations. In particular, assigning arbitrary costs requires analyzing the identified alignments for validity. Notice that the average computation time increases significantly in comparison with the one of **Exp. 1**.

In the third group of experiments (**Exp. 3**), we consider cases where we have background knowledge on the activities that can be swapped/replaced. We set the cost of allowed replacements and swaps to 1, while the cost of replacements and swaps for other pairs of activities was set to $+\infty$. The cost of moves on log (INS) and moves on model (SUP) was set to 1. The results are presented in the last block of Table I. These results show that our technique is able to detect all deviations as the last block coincides with the first block in the table. Moreover, the computation time decreases significantly compared to **Exp. 2**. This shows that with proper knowledge

Trace					Exp. 1			Exp. 2					Exp. 3				
ID	INS	SUP	REP	SWA	INS	SUP	TC	INS	SUP	REP	SWA	TC	INS	SUP	REP	SWA	TC
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	3	0	0	0	3	0	3	3	0	0	0	3	3	0	0	0	3
3	0	2	0	0	0	2	2	0	2	0	0	2	0	2	0	0	2
4	1	2	0	0	1	2	3	1	2	0	0	3	1	2	0	0	3
5	0	0	2	0	2	2	4	0	0	2	0	2	0	0	2	0	2
6	0	0	2	0	1	2	3	0	0	2	0	2	0	0	2	0	2
7	1	2	2	0	6	1	7	1	2	2	0	5	1	2	2	0	5
8	0	0	0	1	1	1	2	0	0	0	1	1	0	0	0	1	1
9	0	0	0	2	2	2	4	0	0	0	2	2	0	0	0	2	2
10	0	1	0	1	1	2	3	0	1	0	1	2	0	1	0	1	2
11	4	1	0	1	5	2	7	3	0	2	0	5	4	1	0	1	6
12	0	0	1	1	2	2	4	0	0	1	1	2	0	0	1	1	2
13	0	0	1	1	3	3	6	0	0	1	1	2	0	0	1	1	2
14	0	2	1	1	5	1	6	0	2	1	1	4	0	2	1	1	4
Average calculation time					0.26ms/trace			114.51ms/trace					0.26ms/trace				

Table I. DEVIATION DETECTION FOR EXPERIMENTS ON SYNTHETIC DATA

about activities that can be replaced/swapped, one can set the cost of deviations properly to get a meaningful analysis as well as reducing computation time.

We also evaluated the approach using real-life logs and models taken from a Dutch academic hospital. Experiment results show that the approach managed to identify similar deviations as the ones identified manually by experts. Due to space constraints, details of the experiments are omitted.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented a flexible framework for controlled break-the-glass based on alignments. The framework allows users to deviate from the specification within a given range. The severity of deviations is determined using alignment-based online and offline conformance checking. Online conformance checking analyzes user behavior at run time. This verification, however, may underestimate the severity of deviations when the process execution does not reach proper termination. Offline conformance checking addresses this issue by computing an optimal alignment between traces in the event log and complete runs of the process model. Moreover, we extend the alignment approach to explicitly show high level deviations, i.e. replacements and swaps. The experiment results show that the proposed approach is robust in the sense that it is able to identify and quantify deviations without proper knowledge about deviations. Prior knowledge on possible deviations can be used to tune the cost function to improve the quality of deviation diagnostics and computation performance.

The work presented in this paper suggests some interesting directions for future work. The proposed approach determines optimal alignments only based on the control flow defined in the process model. Other information like the users responsible for the deviations and the data that have been accessed as well as information about the context may be considered in order to make a security decision. Moreover, a systematic way for determining the deviation budget is needed. To this end, we are investigating methods that assess the risks caused by the invocation of the break-the glass functionality based on historical data.

Acknowledgments. This work has been funded by the Dutch program COMMIT under the THeCS project, by the Cyber Security program under the PriCE project, and by the EU Commission under the Seventh Framework Project "PoSecCo".

REFERENCES

[1] K. W. Hamlen, G. Morrisett, and F. B. Schneider, "Computability classes for enforcement mechanisms," *TOPLAS*, vol. 28, no. 1, pp. 175–205, 2006.

[2] J. Ligatti, L. Bauer, and D. Walker, "Run-time enforcement of nonsafety policies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 1–41, 2009.

[3] A. Ferreira, R. Cruz-Correia, L. Antunes, P. Farinha, E. Oliveira-Palhares, D. W. Chadwick, and A. Costa-Pereira, "How to break access control in a controlled manner," in *Proc. of Symp. on Computer- Based Medical Systems*. IEEE, 2006, pp. 847–854.

[4] S. R. Reti, H. J. Feldman, S. E. Ross, and C. Safran, "Improving personal health records for patient-centered care," *JAMIA*, vol. 17, no. 2, pp. 192–195, 2010.

[5] C. A. Ardagna, S. De Capitani di Vimercati, T. Grandison, S. Jajodia, and P. Samarati, "Regulating exceptions in healthcare using policy spaces," in *Data and Applications Security*, ser. LNCS 5094. Springer, 2008, pp. 254–267.

[6] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari, "Declarative specification and verification of service choreographiess," *TWEB*, vol. 4, no. 1, pp. 3:1–3:62, 2010.

[7] M. Petkovic, D. Prandi, and N. Zannone, "Purpose control: Did you process the data for the intended purpose?" in *Secure Data Management*, ser. LNCS 6933. Springer, 2011, pp. 145–168.

[8] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance Checking Using Cost-Based Fitness Analysis," in *Proc. of Int. Enterprise Distributed Object Computing Conf.* IEEE, 2011, pp. 55–64.

[9] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying History on Process Models for Conformance Checking and Performance Analysis," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[10] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Memory-Efficient Alignment of Observed and Modeled Behavior," BPMcenter.org, Tech. Rep., 2013, bPM Center Report BPM-03-03.

[11] B. Depaire, J. Swinnen, Mieke, and K. Vanhoof, "A Process Deviation Analysis Framework," in *Business Process Management Workshops*, ser. LNBIP 132. Springer, 2013, pp. 701–706.

[12] S. Banescu, M. Petkovic, and N. Zannone, "Measuring privacy compliance using fitness metrics," in *Business Process Management*, ser. LNCS 7481. Springer, 2012, pp. 114–119.

[13] A. Adriansyah, B. F. van Dongen, and N. Zannone, "Controlling break-the-glass through alignment," *ASE SCIENCE Journal*, 2013, To appear.