Controlling break-the-glass through alignment

Citation for published version (APA):

Adriansyah, A., Dongen, van, B. F., & Zannone, N. (2013). Controlling break-the-glass through alignment. *Advanced Science Express*, 2(4), 198-212.

Document status and date:

Published: 01/01/2013

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Controlling Break-The-Glass Through Alignment¹

Arya Adriansyah
Eindhoven University of
Technology
Email: a.adriansyah@tue.nl

Boudewijn F. van Dongen
Eindhoven University of
Technology
Email: b.f.v.dongen@tue.nl

Nicola Zannone
Eindhoven University of
Technology
Email: 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. This causes organizations to employ functionality like break-the-glass that allows users to bypass security mechanisms in case of emergencies. However, break-the-glass introduces a weak point in the system and can be misused. 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. To measure the severity of deviations, we extend alignment-based deviation analysis techniques. In particular, our technique is able to identify high-level deviations such as activity replacements and swaps; hence it provides a more accurate diagnosis of deviations than classical optimal alignments. Our work is implemented as a ProM 6 plug-in and evaluated using both synthetic and real-life data.

I INTRODUCTION

Recent studies show that insider abuse is a significant problem for organizations. According to the 2008 CSI Computer Crime and Security Survey [2], 44% of reported incidents belong to the category of insider abuse, second only to virus incidents (50%). Moreover, damage caused by insiders can be particularly expensive: a US Secret Service study shows that more than 50% of respondents on insider frauds claimed losses over \$20.000 with 12 respondents claimed a loss over \$1.0 million.

The underlying fundamental problem is that nowadays policy specification and enforcement, which guarantee that data are used and distributed according to established policies, are done exclusively by preventive means. 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 [3,4].

Preventive security mechanisms are 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. For instance, doctors may deviate from the specification to react to an emergency. Preventing such actions can result in undesirable consequences (e.g., the loss of a patient).

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 [5,6]. The deployment of break-the-glass functionality, however, introduces a weak point in the system that can be misused [7]. 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 [8,9] 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 [10] provides a robust approach to determine the root causes of deviations. Intuitively, alignments are used to identify and measure the differences between a process execution and a process model by comparing the process execution against the possible runs of the process model. However, constructing optimal alignments is computationally expensive. Existing techniques [11,12] impose restrictions on the type of optimal alignments to compute them efficiently. In particular, alignments only identify events that occur in the process execution although they not allowed according to the process model (i.e., insertions) and activities that must

¹This paper is an expanded version of [1].

occur according to the process model but they are absent in the process execution (i.e., suppressions). Nonetheless, the severity of deviations should be assessed in terms of high level deviations like replacements and swaps [13]. 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 [14].

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. Assuming that all activities can be replaced/swapped with all other activities, however, may reveal misleading diagnostic information and unnecessarily increase computation complexity. Therefore, we also provide guidelines to obtain accurate diagnostic information while preserving efficiency.

Our technique has been implemented as a ProM 6 plug-in and evaluated using both synthetic and real-life logs. In particular, we use synthetic data to assess the ability of the approach to detect deviations. Real-life logs from a Dutch Hospital are used to evaluate

the approach for controlled break-the-glass functionality.

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 experimental results. Finally, Section VI discusses related work, and Section VII concludes the paper providing directions for future work.

II PRELIMINARIES

This section introduces the preliminary concepts used in the remainder of the paper.

Process models describe how processes should be carried out. We consider process models in the form of Petri nets [15]. Our approach is extendable to any modeling language for which a translation to Petri nets is available. 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.

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

Figure 1: Example of treatment process

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 Figure 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 Figure 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. Figure 2 shows some examples of traces for the process in Figure 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 Figure 2).

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

Figure 2: Example of traces for the model in Figure 1

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. Take for example the net in Figure 1 and its perfectly fitting trace σ_1 in Figure 2. For every occurrence of an activity in the trace, there is a transition in the net mapped to the same activity that can be fired. Moreover, after all activities in the trace are mimicked, the net reaches its final marking.

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 $\gg$,
- (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 Figure 1 and trace σ_2 in Figure 2. γ_1 in Figure 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. As shown in γ_1 , the sequence of activities in the top row yields the original trace σ_2 and the sequence of transitions in the bottom row is a complete run of the net.

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

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

Figure 3: An alignment (top) and a prefix alignment (bottom) between trace σ_2 in Figure 2 and the model in Figure 1

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 Figure 3 is a prefix alignment between the net in Figure 1 and trace σ_2 in Figure 2. Notice that the top row of the alignment yields trace σ_2 (the same as γ_1 in Figure 3), but the bottom row yields a prefix of a complete run.

For a given process model and a trace, there can be more than one possible (prefix) alignment. For instance, γ_3 in Figure 4 is another possible alignment between trace σ_2 and the net in Figure 1. Notice that there are more deviations according to γ_3 than according to γ_1 (4 compared to 3).

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

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

The quality of a (prefix) alignment is measured based on a predefined *cost function*. A cost function defines the cost of movements that can occur in the alignment. 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 [16] defines the cost of each move on model/move on log equal to 1, 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. Given a Petri net and a trace, the complexity of computing optimal alignments between the trace and the net is exponential in the number of transitions in the net and the number of activities in the trace. Existing approaches (e.g., [11,12]) exploit structural properties of both Petri nets and traces to construct optimal alignments efficiently. In particular, to increase efficiency these approaches only allow synchronous moves between an activity and a transition labeled with the same activity.

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 Figure 5. In the remainder of the section, we introduce the basic concepts and provide an overview of the architecture.

Figure 5: Controlled Break-the-glass 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 completely spent, 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 functionality is invoked, is used to determine the initial marking on the process model. The conformance checker determines the cost-optimal alignment starting from this initial 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 the deviations in the log and quantify them. However, online conformance checking may underestimate the total cost of deviations if the process execution ends without reaching proper termination according to the model. 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, which is reported to the security officer.

Take for example the process in Figure 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 Figure 6 (left). After the user invoked break-the-glass, he performs **Lab test**. Figure 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.

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

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 Figure 6(ii). However, after the occurrence of **Evaluation**, conformance checking returns an optimal prefix alignment with deviation cost of 2 (Figure 6(iii)). The allocated deviation budget is overspent. Thus, the exceptional permissions are revoked and the security officer is informed about the infringement.

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 Figure 6. The optimal alignment has cost of 3 (a move on model for t_9 (**Home treatment**) is added to the alignment in Figure 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 Figure 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 [14]. 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

The extension to identify replacements and swaps are explained in Sections 1 and 2 respectively. Section 3 provides guidelines for setting the cost of these deviations.

1 REPLACEMENT

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 Figure 1 and trace σ_3 in Figure 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 Figure 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 Figure 7. Here, the synchronous move between activity **Therapy** and transition t_9 (**Home treatment**) shows the *replacement* of activity **Home treatment** with activity **Therapy**.

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|} \hline a & r & l & \gg & e & l & t \\ \hline a & r & l & c & e & & h \\ \hline t_1 & t_2 & t_3 & t_4 & t_5 & \gg & t_9 \\ \hline \end{array}$$

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

To efficiently compute optimal alignments, existing approaches (e.g., [11, 12]) cannot construct optimal alignments that show replacements such as γ_3 be-

cause 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*. Figure 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.

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

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 Figure 2 and the net of Figure 1 augmented with the substitution transition in Figure 8, using the extended cost function derived from δ_1 , is shown in Figure 9.

$$\gamma_5 = \begin{array}{|c|c|c|c|c|c|c|} \hline a & r & l & \gg & e & l & t \\ \hline a & r & l & c & e & & t \text{ (replacing h)} \\ \hline t_1 & t_2 & t_3 & t_4 & t_5 & \gg & t'_9 \\ \hline \end{array}$$

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

Translating the deviations in γ_5 (Figure 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 Figure 7.

The proposed solution comes with a price of computation complexity. Suppose that the trace consists of n unique activities, then its extended net with substitution transitions has n times transitions more than the original net. Thus, the use of extended nets increases the computation complexity of computing optimal alignments by a constant factor n .

2 SWAPPED ACTIVITIES

A deviation of type insertion, suppression, or replacement manifests in alignments as an individual movement, i.e. a move on log, a move on model, or a synchronous move; in contrast, *swaps* are defined over pairs of activities. Hence, they cannot be directly identified from individual movements in optimal alignments. Consider again the model in Figure 1, trace σ_2 in Figure 2, and their alignment γ_1 in Figure 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. Similarly to the approach for identifying replacements, 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* $t_{2 \rightarrow 1}$ and $t'_{2 \rightarrow 1}$ respectively and reverse their label. Moreover, a place $p_{t_2 \rightarrow t_1}$ is added to connect $t_{2 \rightarrow 1}$ and $t'_{2 \rightarrow 1}$ such that proper completion on the model is reached if and only if either both transitions are fired or none of them are fired. Figure 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 $t_{3 \rightarrow 2}$ must be followed by firing of $t'_{3 \rightarrow 2}$, otherwise there is a remaining token in place $p_{t_3 \rightarrow t_2}$ which implies that proper termination is not

reached.

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

We extend the original cost function for insertions and suppressions as follows. The cost of swapping two activities is distributed equally among the corresponding interchange transitions. Note that interchange transitions are not part of the original model and, thus, they cannot be considered independently in alignments. To this end, 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.

Consider again the net in Figure 1 and trace σ_2 in Figure 2 where the cost of insertions/suppressions follows the standard cost function, the cost of swapping activities **Radiology** with **Lab test** is 1, and the cost is $+\infty$ otherwise. An optimal alignment between σ_2 and the net augmented with the interchange transitions in Figure 10, is shown in Figure 11.

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

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

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

Note that prefix alignments show an optimistic view of deviations with respect to swapped activities, i.e. an optimal prefix alignment may assume that swaps will eventually occur in the future. For example, us-

ing the same cost function as the one used in the construction of γ_6 , the alignment γ_7 in Figure 12 is an optimal prefix alignment between trace $\langle a, l \rangle$ and the model in Figure 1. The synchronous move between activity **Lab test** and interchange transition $t_{3 \rightarrow 2}$ suggests that activity **Radiology** is swapped with activity **Lab test**, although no event in the trace refers to activity **Radiology**.

$$\gamma_7 = \begin{array}{c|c} a & l \\ \hline a & l \text{ (r swapped with l)} \\ t_1 & t_{3 \rightarrow 2} \end{array}$$

Figure 12: An optimal prefix alignment, showing optimistic interpretation on deviations

The idea of interchange transitions can be generalized to identify swaps of sequences of activities. Given a process model and a trace, for all pairs of sequences of activities μ and μ' that can be swapped (i.e., μ' can occur first instead of μ while the original model shows the other way around), we duplicate all transitions in the model that correspond to activities in both sequences. Places are added between duplicate transitions of μ' and μ according to the ordering of their activities in the original sequences. Then, we swap the ordering between the sequences of duplicate transitions μ' and μ and add a place between the last duplicate transition of μ' and the first duplicate transition of μ . By augmenting the model with such constructs, optimal (prefix) alignment between the extended net and the original trace may reveal swapped sequences of activities in the trace. The sketch of this generalized approach is shown in Figure 13.

Figure 13: Sketch of solutions for the identification of swaps between two sequences of activities

Given a trace and a Petri net, in the worst case where all activities in the trace can be swapped with one another, the number of possible swaps is $n(n-1)$, where n is the number of unique activities in the trace. If we allow sequences of activities to be swapped, the com-

plexity is hyper-exponential in the number of activities because the number of possible sequences grows exponentially in the number of their elements.

3 GUIDELINES FOR SETTING COST FUNCTION

To identify all types of deviations at once, i.e. insertion, suppression, replacement, and swap, a Petri net can be augmented with both interchange and substitution transitions. The cost of movements involving these transitions is defined as shown in Sections 1 and 2. Note that our approach ensures that an event in the trace is involved in at most one type of deviation, e.g. if an occurrence of an activity is swapped with another activity according to an optimal alignment, the same occurrence cannot be identified as inserted or swapped with another activity. Thus, our approach provides clarity in deviation analysis that can be exploited for further analysis.

So far, we assume that the costs of replacements and swaps are given. However, defining such costs is often not trivial in practice. For instance, a naïve extension of the standard cost function for replacements and swaps would assign a cost equal to 1 to every replacement and swap. This function, however, causes an increase of the complexity of computing optimal alignments because all replacements and swaps between activities need to be explicitly modeled. Furthermore, such a function may yield misleading deviation diagnostics. For instance, in the net of Figure 8, allowing the replacement of **Home Treatment** with **Appointment** can provide misleading diagnostic information, as in reality they are two significantly different activities. To reduce the complexity while providing accurate diagnosis, substitution transitions and interchange transitions should be added only when they provide intuitive deviation analysis.

Possible replacements should be determined by analyzing the activities in the process model and event log. In particular, it is necessary to identify which activities can be replaced by other activities in the process execution and estimate the severity of such replacements. The process for the identification of replacements and estimation of their costs can be supported by the use of metrics for assessing the similarity degree between activities. For instance, the work in [17] uses the Latent Semantic Analysis (LSA) [18], a semantic relatedness metric, to assess the similarity between two activities based on their name. Ontology alignment techniques [19,20] can be also employed to

assess the similarity degree between two activities. The application of these techniques requires representing activities as concepts in an ontology. Ontology alignment techniques then can compute the degree of semantic resemblance between concepts based on the ontology structure. The cost of replacements can be defined as a function of the degree of similarity: higher the similarity degree between two activities, lower it is the cost of replacing one activity with the other.

Swaps can be determined by analyzing the activities and control flow in the process model. The control flow defines a precise ordering in which activities should be performed. In some case, when the swap of two activities has an insignificant or limited impact on the process execution, the constraints imposed by the control flow can be relaxed (e.g., activities *Radiology* and *Lab test* in Figure 1). However, not every swap may be allowable. In particular, an activity cannot be swapped with another activity if the execution of the latter is required for the execution of the former (e.g., in Figure 1 *Evaluation* is required for *Operation*). We call such a pair of activities *unswappable activities* and write $(\text{Evaluation}, \text{Operation})$ to indicate that *Evaluation* cannot be swapped with activity *Operation*.

The set of unswappable activities specifies the basic pairs of activities that cannot be swapped. The constraints imposed by this set may entail constraints on other pairs of activities. For instance, an activity should not be swapped with the predecessors of an activity whose execution is necessary for the execution of the former activity. Indeed, allowing such a swap will result in an undesirable alignment. Consider, for instance, the net in Figure 1 and the trace $\langle \text{Appointment}, \text{Radiology}, \text{Operation}, \text{Check history}, \text{Evaluation}, \text{Lab test}, \text{Nursing ward} \rangle$. If *Lab test* is swapped with *Operation*, this would allow *Operation* *before* an *Evaluation*.

To prevent the approach from identifying misleading swapping, we define the *unswappable activity closure*. Given a Petri net and a pair of unswappable activities (a, b) , the closure of (a, b) consists of the set of pairs of activities (c, b) such that c is a predecessor of a in the net and the set of pairs of activities (a, d) such that d is a successor of b in the net. For instance, the closure of $(\text{Evaluation}, \text{Operation})$ with respect to the net in Figure 1 is the set $\{(\text{Appointment}, \text{Operation}), (\text{Radiology}, \text{Operation}), (\text{Lab test}, \text{Operation}), (\text{Check history}, \text{Operation})\} \cup \{(\text{Evaluation}, \text{Nursing ward})\}$. The costs of pairs of activities in the unswappable

activity closure is set to infinitely large $(+\infty)$. A pair of activities can be swapped only if, given a set of pairs of unswappable activities S , such a pair does not belong the unswappable activity closure of S . Its cost should be proportional to its impact on the process execution.

We stress that only replacements and swaps whose cost is lower than the sum of the costs of the moves on model and moves on log forming such replacements and swaps need to be considered, while the cost of other deviations can be safely assumed to be infinitely high. If the cost of a replacement or a swap is greater than that sum, the replacement or the swap will never be considered in the alignment.

V EXPERIMENTS

We have implemented the proposed conformance checking technique as a ProM 6 plug-in, publicly available at www.processmining.org. We performed a number of experiments to demonstrate the conformance checking technique using both synthetic data and real data logs. In particular, we evaluate offline conformance checking, i.e. we assume that traces of execution have reached proper termination, as it provides non-optimistic deviation diagnostics. The experiments were performed using an Intel Core i7 processor (2.80 GHz) with 8GB RAM.

1 SYNTHETIC DATA

The first set of experiments aims to evaluate the robustness of the approach and, in particular, its ability to detect deviations. 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 1). Three groups of experiments were performed. For each experiment, we report the identified deviations and the total cost for the optimal alignment (TC). For each group of experiments, we also report the average calculation time per trace.

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 1: Deviation detection for experiments on synthetic data

In the first group of experiments (**Exp. 1**), we use existing alignment techniques (e.g., [21]) 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 1. 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 1). 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 1.

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**. This increment confirms the exponential complexity introduced when considering all possible replacements and swaps.

Figure 14: Experiment results based on real-life log and model

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 1. 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 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. Note that without such knowledge, one can perform an iterative procedure to obtain well-founded results by first using the standard cost function as we did for **Exp. 2**, and then setting the cost for identified incompatible swapped/replaced activities to high (i.e., $+\infty$) in the next iterations.

2 REAL-LIFE LOGS

To evaluate the applicability of the approach in a real scenario, we use a real-life event log from a Dutch academic hospital which has been made available for the BPI Challenge 2011 [22]. Since no process model is available for the log, a model was mined from the log. The original log contains cases referring to different diagnoses (i.e., different treatment processes), which makes the log to be *heterogeneous*. Mining heterogeneous logs, however, may lead to inaccurate process models [23]. We considered only cases related to diagnosis for cancer of the vulva. The obtained log contains 493 events in 94 cases. 70% of these cases (i.e., 66 cases) were used to mine the process model using the *Alpha Miner* ProM plug-ins [24]. For the purpose of the experiments, we assumed that the mined process model provides the correct representation of the

selected diagnosis. This model was analyzed to determine the costs of deviations. In particular, we analyzed the activities in the model and the control flow to identify which activities can be replaced by other activities (e.g., similar medical tests or procedures) as well as which activities can be swapped without a significant impact on the process execution. A cost for such deviations was defined on the basis of their severity. The remaining 30% of cases (i.e., 28 cases) was manually analyzed to evaluate the output of the plug-in. In particular, we analyzed the events executed in each case and compared them with the control flow specified in the process model. The analyzed cases were assessed using a five-scale system (i.e., *low*, *low/medium*, *medium*, *medium/high*, and *high*); each scale represents the overall severity of the deviations in a case.

The cases that were analyzed manually together with the defined cost function were given as input to the plug-in. The average calculation time per trace was 115.53 ms. The results are presented in Figure 14. As shown in the figure, the results of the plug-in coincide with the ones obtained through manual analysis in 16 cases (out of the 28 cases analyzed). For the other 12 cases, there is a slight difference between the results of the plug-in and the ones of manual analysis. Here, we analyze and discuss the reasons behind such differences. A summary is presented in Table 2.

<i>n. cases</i>	<i>Result</i>	<i>Motivation</i>
1	Overestimation	Repetition of the same transition
6	Overestimation	Reordering
5	Underestimation	Difficult to manually identify a corresponding run of the process model

Table 2: Analysis of mismatches between manual and automated investigation

The plug-in overestimates the results of manual analysis in seven cases and underestimate them in five

cases. In one case (#8), an activity, which is allowed by the specification, was repeated several times. Although these deviations were correctly identified by the plug-in as moves on log, the plug-in does not discriminate which activities were inserted. In contrast, the repetition of the same activity was considered as a mitigating factor during manual analysis.

In six cases, the plug-in overestimates the results of manual analysis because the swap of sequences of activities is not supported by the plug-in. An analysis of these cases showed that in two cases (#784 and #793) the alignment identified through manual inspection and by the plug-in was exactly the same; the higher cost is caused by the fact that swaps of sequences of activities were accounted as combinations of moves on log and moves on model. In the other cases (#68, #132, #546, and #783), the plug-in aligned the log with a run of the process model which was different from the one indicated by manual analysis. Note that, although our implementation only considers swaps of two activities, our approach is general enough to deal with swaps of sequences of activities (see Section IV).

The cases in which the plug-in returned a severity level lower than the level indicated by manual analysis correspond to cases for which it was difficult to identify an alignment. As a consequence, the severity of these cases might have been overestimated in the manual assessment. In three cases (#73, #344 and #1048), the alignment determined by the plug-in was reliable. This demonstrates that the plug-in can help in understanding the root causes of deviations. In the other two cases (#92 and #1110) the plug-in aligned the cases with very short run of the process model and accounted most events in the log as moves on log. This is reflected in the *move-log fitness* metric of alignments, i.e. the ratio between the total cost of synchronous moves in the alignment and the total cost obtained by considering all activities in the trace as moves on log. This metric provides an intuition on how much the activities in the trace are deviating from the model. In the last two cases, this metric value is very low (below 0.14 from scale of 0.0 to 1.0). This indicates that the execution of activities in the traces is rather random and does not follow the flow of activities described in the model.

VI RELATED WORK

Little work in the literature considers how to incorporate break-the-glass functionality in existing infras-

tructures. For instance, Brucker and Petritsch [25] present a break-the-glass enforcement architecture which extend access control models with the set of emergency policies. This approach, however, is not practical as it requires identifying all possible exceptions at design time. Ferreira et al. [5] extend the access control system with a break-the-glass policy that can be invoked in case of emergency. Although such a policy is coupled with logging mechanisms that record users' actions and non-repudiation mechanisms, methods for the analysis of user behavior are not discussed. In contrast, we presented a flexible architecture that allows users to invoke the break-the-glass functionality at any time and alerts when deviations from specifications cannot be tolerated.

Many approaches have been proposed to check recorded process executions against predefined rules/process models. van der Aalst et al. [26] proposed an LTL-based approach to check the satisfiability of a set of properties in a given trace. In this approach, the satisfiability of each property is checked independently. Hence, correlations between deviations in one rule and other deviations in other rules cannot be easily identified. An approach based on the behavioral profile is proposed in [27]. Given a set of traces and a process model, this approach encodes the behavior of both the model and traces in metric representations before comparison. Although the approach supports root-cause analysis, the metric representation may abstract away behavior that is important for analysis. Banescu et al. [14] present an approach for identifying and measuring privacy deviations based on token-based replay techniques [28]. The genetic mining algorithm in [29] uses similar replay technique to measure the quality of process models with respect to given executions. Token-based replay techniques, however, may allow behavior that is not allowed by the model due to the used heuristic and thus may provide misleading diagnostic information [30].

The notion of alignments [10] provides a robust approach to correlate occurrences of recorded activities with transitions in predefined process models using their semantics. In particular, the notion of alignments is based on the "edit distance" of a trace from a process model. This idea has been used to extend the theory of runtime enforcement by introducing the notion of predictability [31]. In particular, edit distance metrics are used to map "almost valid" traces into the predictable (closest valid) trace. In [17] edit distance metrics have been enhanced to measure the privacy distance between expected and actual user

behavior. The problem of these metrics is that they cannot be applied to analyze logs against complex process models, for instance including loops. Moreover, constructing alignments is computationally expensive. Adriansyah et al. [11, 12, 21] propose efficient approaches to construct optimal alignments by restricting the map of process executions to process models, and thus improve the applicability of techniques based on alignments to handle real-life cases. Nevertheless, these approaches only identify low level deviations (i.e., moves on log and moves on model). This may lead to misinterpret the root causes of deviations when low level deviations are analyzed to identify high level deviations. Our approach extends the work in [11, 12, 21] by allowing the identification of high level deviations through alignment, while preserving efficiency.

VII 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 op-

timal 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 in order 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 executed activities and 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 to quantify the severity of deviations, leading to more accurate security decisions. 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] Arya Adriansyah, Boudewijn F. van Dongen, and Nicola Zannone, “Controlling Break-The-Glass Through Alignment”, in *Proceedings of International Conference on Information Privacy, Security, Risk and Trust*. 2013, IEEE.
- [2] Robert Richardson, “Computer Crime & Security Survey”, Tech. report, Computer Security Institute, 2008.
- [3] Kevin W. Hamlen, Greg Morrisett, and Fred B. Schneider, “Computability classes for enforcement mechanisms”, *TOPLAS*, vol. 28, no. 1, pp. 175–205, 2006.
- [4] Jay Ligatti, Lujo Bauer, and David Walker, “Run-time enforcement of nonsafety policies”, *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 1–41, 2009.
- [5] 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 *Proceedings of Symposium on Computer- Based Medical Systems*. 2006, pp. 847–854, IEEE.
- [6] Shane R Reti, Henry J Feldman, Stephen E Ross, and Charles Safran, “Improving personal health records for patient-centered care”, *JAMIA*, vol. 17, no. 2, pp. 192–195, 2010.
- [7] Claudio Agostino Ardagna, Sabrina De Capitani di Vimercati, Tyrone Grandison, Sushil Jajodia, and Pierangela Samarati, “Regulating exceptions in healthcare using policy spaces”, in *Data and Applications Security*. 2008, LNCS 5094, pp. 254–267, Springer.
- [8] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello, and Sergio Storari, “Declarative specification and verification of service choreographies”, *TWEB*, vol. 4, no. 1, pp. 3:1–3:62, 2010.

- [9] Milan Petkovic, Davide Prandi, and Nicola Zannone, "Purpose control: Did you process the data for the intended purpose?", in *Secure Data Management*. 2011, LNCS 6933, pp. 145–168, Springer.
- [10] J. E. Cook and A. L. Wolf, "Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model", *TOSEM*, vol. 8, pp. 147–176, 1999.
- [11] A. Adriansyah, N. Sidorova, and B. F. van Dongen, "Cost-Based Fitness in Conformance Checking", in *Proceedings of International Conference on Application of Concurrency to System Design*. 2011, pp. 57–66, IEEE.
- [12] A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst, "Memory-Efficient Alignment of Observed and Modeled Behavior", Tech. Rep., BPMcenter.org, 2013, BPM Center Report BPM-03-03.
- [13] Benoit Depaire, Jo Swinnen, Mieke, and Koen Vanhoof, "A Process Deviation Analysis Framework", in *Business Process Management Workshops*. 2013, LNBIP 132, pp. 701–706, Springer.
- [14] Sebastian Banescu, Milan Petkovic, and Nicola Zannone, "Measuring privacy compliance using fitness metrics", in *Business Process Management*. 2012, LNCS 7481, pp. 114–119, Springer.
- [15] T. Murata, "Petri nets: Properties, analysis and applications", *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 2002.
- [16] 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.
- [17] Sebastian Banescu and Nicola Zannone, "Measuring privacy compliance with process specifications", in *Proceedings of International Workshop on Security Measurements and Metrics*. 2011, pp. 41–50, IEEE.
- [18] Susan T. Dumais, "Latent semantic analysis", *ARIST*, vol. 38, no. 1, pp. 188–230, 2004.
- [19] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy, "Learning to match ontologies on the Semantic Web", *VLDB Journal*, vol. 12, no. 4, pp. 303–319, 2003.
- [20] Steven Heeps, Joe Sventek, Naranker Dulay, Alberto Schaeffer Filho, Emil Lupu, Morris Sloman, and Stephen Strowes, "Dynamic Ontology Mapping for Interacting Autonomous Systems", in *Proceedings of International Workshop on Self-Organizing Systems*. 2007, LNCS 4725, pp. 255–263, Springer.
- [21] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance Checking Using Cost-Based Fitness Analysis", in *Proceedings of International Enterprise Distributed Object Computing Conference*. 2011, pp. 55–64, IEEE.
- [22] B.F. van Dongen, "Event Log for the BPI Challenge 2012", <http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>, 2012.
- [23] R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst, "Context aware trace clustering: Towards improving process mining results", in *Proceedings of International Conference on Data Mining*. 2009, pp. 401–412, SIAM.
- [24] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster, "Workflow Mining: Discovering Process Models from Event Logs", *TKDE*, vol. 16, no. 9, pp. 1128–1142, 2004.
- [25] Achim D. Brucker and Helmut Petritsch, "Extending access control models with break-glass", in *Proceedings of Symposium on Access Control Models and Technologies*. 2009, pp. 197–206, ACM.
- [26] Wil M. P. van der Aalst, H. T. de Beer, and Boudewijn F. van Dongen, "Process mining and verification of properties: An approach based on temporal logic", in *On the Move to Meaningful Internet Systems*. 2005, LNCS 3760, pp. 130–147, Springer.
- [27] Matthias Weidlich, Artem Polyvyanyy, Nirmal Desai, Jan Mendling, and Mathias Weske, "Process compliance analysis based on behavioural profiles", *Inf. Sys.*, vol. 36, no. 7, pp. 1009–1025, 2011.
- [28] Anne Rozinat and Wil M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior", *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.
- [29] A. K. Alves de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic Process Mining: an Experimental Evaluation", *Data Mining and Knowledge Discovery*, vol. 14, no. 2, pp. 245–304, 2007.
- [30] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Towards Robust Conformance Checking", in *Business Process Manage-*

- ment Workshops*. 2011, LNBIP 66, pp. 122–133, Springer.
- [31] Nataliia Bielova and Fabio Massacci, “Pre-dictability of enforcement”, in *Engineering Secure Software and Systems*. 2011, LNCS 6542, pp. 73–86, Springer.