LETTER

Temporal Outlier Detection and Correlation Analysis of Business Process Executions

Chun Gun PARK†, Nonmember and Hyun AHN††, Member

SUMMARY  Temporal behavior is a primary aspect of business process executions. Herein, we propose a temporal outlier detection and analysis method for business processes. Particularly, the method performs correlation analysis between the execution times of traces and activities to determine the type of activities that significantly influence the anomalous temporal behavior of a trace. To this end, we describe the modeling of temporal behaviors considering different control-flow patterns of business processes. Further, an execution time matrix with execution times of activities in all traces is constructed by using the event logs. Based on this matrix, we perform temporal outlier detection and correlation-based analysis.

key words: process-aware information systems, temporal outlier detection, execution time, correlation analysis

1. Introduction

In managing modern business processes, it is vital to monitor, measure, and control the business process executions in real-time. Particularly, anomalous behaviors observable during business process executions should be managed effectively because they can contribute significantly to undermining business process performance resulting in failures. To this end, several studies have been conducted to identify the causes and characteristics of abnormal behaviors through post-analysis using event logs generated by process-aware information systems. Further, predictive monitoring techniques [1], which aim to detect and avoid undesirable behaviors in the running processes in advance, have been introduced in the literature.

In this regard, this study mainly focuses on the temporal aspect of business processes. Basically, a business process consists of a number of activities. Each of these activities requires a variable amount of execution time, and the total execution time of all activities is directly related to the time performance of the business process. Therefore, in a time-critical business process, the execution of a particular activity is considered to be an exceptional case if it is terminated prematurely or requires considerably longer time than expected. As such, anomalous temporal behaviors of business processes have been analyzed in a few studies [2], [3]. This study attempts to detect and analyze temporal outliers in business process executions. Compared to [2], which only considered temporal behaviors of activity executions, our method is to analyze the correlations between activities and traces. In other words, our method is to quantify how much each activity type affects the occurrences of temporal outlier through the correlation analysis between execution times at trace- and activity-levels. To this end, we first divide all the traces into several groups with the same execution path. Further, for each trace group, we measure the execution times of all activities by considering the control-flow patterns of the business process (sequential, parallel, and iterative patterns). Finally, our method performs a correlation analysis between execution times of traces and activities, and the analyzed result is visualized as a parallel plot and correlation matrix.

2. Modeling Temporal Behaviors of Business Process Executions

Execution time typically represents temporal behaviors of a business process and can be measured from event logs generated from the business process executions. As preliminaries, the event log and its subordinate concepts are formally represented as follows:

Definition 1 (Event Log). An event log \( \mathcal{L} \) is formally defined as a 6-tuple \( \mathcal{L} = (A, \mathcal{C}, \mathcal{E}, \chi, \delta, \tau) \), where \( A \) is a set of activities, \( \mathcal{C} \) is a set of traces (cases), \( \mathcal{E} \) is a set of events, \( \chi : \mathcal{E} \rightarrow \mathcal{C} \) is a function from an event \( e \in \mathcal{E} \) to the trace including \( e \), \( \delta : \mathcal{E} \rightarrow A \) is a function from an event \( e \) to its corresponding activity, and \( \tau : \mathcal{E} \rightarrow t_j \) is a function from an event \( e_i \) to its timestamp information \( t_j \).

An event log \( \mathcal{L} \) pertains to a process model that consists of activities \( (A = a_1, a_2, \ldots, a_n) \) and can be executed by process-aware information systems. A trace corresponds to a single execution of the process model (also called a case), therefore the event log is the result of business process executions, and generally contains multiple traces \( (\mathcal{C} = c_1, c_2, \ldots, c_m) \). A trace \( c_i \) has a set of events that have sequentially occurred during the executions of \( (e_{i1}, e_{i2}, \ldots, e_{ik} \subseteq \mathcal{E}) \). An event is generated whenever the running status of the activity changes (e.g., created, started, and completed). In this study, we only handle the completed event type to simplify the event log and measure execution times of individual activities.

To analyze temporal outliers, we measure two types of execution times from event logs: trace-level execution time \( (ET_c) \) and activity-level execution time \( (ET_a) \). The former is obtained by calculating the difference between times-
Table 1 A fragment of the event log

| Trace Id | Activity Type | Timestamp    |
|----------|---------------|--------------|
| A        | ER Triage     | 2014-09-11 09:40:09 |
|          | Leucocytes    | 2014-09-11 10:05:00 |
|          | Admission NC  | 2014-09-11 14:18:24 |
|          | Release A     | 2014-09-20 11:20:00 |
| B        | ER Triage     | 2014-10-21 11:37:27 |
|          | Lactic Acid   | 2014-10-21 11:52:00 |
|          | Release B     | 2014-10-23 00:30:00 |

![Sequential Routing Pattern]

Fig. 1 Execution time measurement for the sequential routing pattern

![Parallel Routing Pattern]

Fig. 2 Execution time measurement for the parallel routing pattern

tamps of the first and last events ($e_{i1}$ and $e_{ik}$) in the trace $c_i$.

$$ET_A(c_i) = \tau(e_{ik}) - \tau(e_{i1})$$

On the contrary, to measure execution times of the latter type, more elaborate considerations must be included according to the control-flow pattern formed among executions of activities in a certain trace.

**Sequential routing pattern.** According to the predefined process model, there are precedence relationships between activities, e.g., every execution of activity $a_1$ always precedes the executions of $a_2$ and $a_3$, which is represented in Fig. 1. Additionally, for the case of $a_2$ and $a_3$, if an activity directly precedes another activity, we denote the relationship as $a_2 \leadsto a_3$. Therefore, the execution time of $a_3$ is the duration between the completion times of $a_3$ and its direct preceding activity $a_2$. Specifically, if there exists a direct precedence relationship between two consequent activities ($a_p \leadsto a_s$) for the trace $c_i$, the execution time of $a_s$ can be measured by calculating the difference between the timestamps of events ($e_p, e_s$) corresponding to the two activities $a_p$ and $a_s$.

$$ET_A(a_s) = \tau(e_s) - \tau(e_p)$$

**Parallel routing pattern.** Figure 2 shows the parallel split/join pattern (denoted as ●) in the process model and its execution time measurements. Activities $a_2$ and $a_3$ are executed independently of each other, and they have direct precedence relationships with $a_1$ ($a_1 \leadsto a_2$ and $a_1 \leadsto a_3$), because they are branched from $a_1$. On the contrary, although $a_3$ has direct precedence relationships with $a_2$ and $a_3$ ($a_2 \leadsto a_3$, $a_3 \leadsto a_4$), the actual execution of $a_4$ starts after both the preceding activities are completed and synchronized. For example, if $a_3$ is completed before $a_3$ in a certain trace, the execution of $a_4$ begins at the completion time of $a_3$ which was completed later than $a_2$.

**Iterative routing pattern.** Handling repetitive activities (placed between the loop enter/exit denoted as ⊕) in a process model is a tricky problem. To simplify the modeling of temporal behavior, we count only the first execution time of each activity as a measurement target, as shown in Fig. 3; in this paper, therefore, we do not take into account subsequent execution times of repetitive activities. This approach is naive, so iterative routing patterns might be addressed with more sophisticated approaches in other works.

Based on the considerations of the above control-flow patterns, an execution time matrix ($X$) for the entire trace is constructed from the event log. The rows and columns of the matrix correspond to traces and activity types, respectively, and each element $x_{ij}$ is the measured value of the execution time of the activity $a_j$ in the trace $c_i$.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} = (x_{ij}) \in \mathbb{R}^{m \times n}$$

A single correlation analysis takes a subset from the matrix by selecting rows corresponding to the traces of the same control-path and quantifies correlations of the measured execution times between each trace and its activities.

3. Temporal Outlier Detection and Analysis

Given that process models tend to include a number of control-flow paths having different execution lengths (numbers of events), it is not reasonable to analyze temporal behaviors on the whole paths in a batch. Therefore, a single correlation analysis is performed on each trace group having the same path. The temporal trace type is a concept for classifying such traces based on their paths.

**Definition 2 (Temporal Trace Type).** A temporal...
trace type is a unique execution sequence of \( k \) ordered activities and is represented as \( \sigma = a_1, a_2, \ldots, a_k \subseteq A \).

In Fig. 4, five temporal trace types are constructed from the process model example represented as an information control net [4]. The traces of event logs are classified by matching the order of events which occur in each trace and the temporal trace types. As mentioned above, the repetitive executions of activities are simplified as a temporal activities order containing only the activity presence/absence information. As shown in Fig. 4, the first temporal trace type, \( \sigma_1 = a_A, a_B, a_F, a_E, a_I \), applies to the traces in which activities \( a_E \) and \( a_F \) in the loop part are executed one or more times. On the contrary, the second type, \( \sigma_2 = a_A, a_B, a_F, a_I \), is matched to the traces where executions of activities in the loop part do not occur. For the parallelized activities (\( a_C \) and \( a_D \)), different trace types are created because they have a combination of execution order.

For the temporal outlier detection, we exploit a robust statistical method [5], which is used to achieve a good performance for data drawn from probability distributions that are not normal. Through the summation of all the measured data of execution time, the probability density functions of \( ETC \) and \( ETA \) of a certain trace group can be obtained empirically.

Let \( Q_1 \) and \( Q_3 \) be the first and third quartiles of the data, respectively. The interquartile range (IQR), which is used to find outliers as a measure, is defined as follows:

\[
IQR = Q_3 - Q_1
\]

As shown in Fig. 5, outliers are classified into mild and extreme types based on the quartiles and IQR. If there exists data \( x_{\text{mild}} \) as a mild outlier, the corresponding range of values is as follows:

\[
Q_1 - 3 \cdot IQR \leq x_{\text{mild}} \leq Q_1 - 1.5 \cdot IQR
\]

The extreme outlier is a more severe type than the mild outlier. If data \( x_{\text{ext}} \) is an extreme outlier, its value range is defined as follows:

\[
x_{\text{ext}} < Q_1 - 3 \cdot IQR \\
\text{or} \\
x_{\text{ext}} > Q_3 + 3 \cdot IQR
\]

On the basis of the definition of outliers, temporal outliers at both levels (trace and activity) can be detected. Furthermore, we apply the Pearson correlation coefficient to quantify correlation relationships among the whole measured data at both levels. In terms of visualization, the parallel coordinates plot and correlation matrix are employed in this study for displaying temporal outlier analysis results.

4. Experiment

To verify our contribution, we conduct an experiment of the temporal outlier detection and analysis. As an experimental dataset, we apply the event logs of a real business process for a medical treatment service called Sepsis Cases [6]. The given event logs pertain to the business process for handling urgent sepsis patients, and all the records included in the logs are stored by the enterprise resource planning (ERP) system of the hospital.

Figure 6 represents the process model discovered by the inductive miner algorithm [7]. The process contains 16 activities, each of which represents an individual medical treatment step for sepsis patients (e.g., triage and CRP measurement) and control-flow constructs such as events, gateways, and arcs. Based on the trace grouping described in Table 2, the event logs are grouped into trace types for further analysis.

Table 2: Details of the event log dataset [6]

| Process name  | Sepsis cases |
|---------------|--------------|
| Start date    | 2013-11-07   |
| End date      | 2015-06-05   |
| Num. of activities | 16         |
| Num. of traces | 1,050        |
| Num. of events | 15,214       |
| Num. of trace types | 472        |
above, we apply the correlation analysis to primary trace types with high frequencies which can be discovered using event logs. Figure 7 represents the trace types $\sigma_{70}, \sigma_{12}, \sigma_{36},$ and $\sigma_{43}$ ranked as the top 4 with their high frequencies of 32, 21, 21, and 20, respectively.

For each trace group, temporal behaviors of all traces which fall into the trace group are visually represented on a parallel coordinates plot as shown in Fig. 8. This visualization method has been widely applied for high dimensional visualization, and consists of parallel axes and polyline, which play a role in each dimensional data intersecting the corresponding axis. The x-axis represents activity types, whereas the y-axis represents measured execution times normalized by using the min-max normalization. In this example, only one trace is detected as the mild outlier (refer to the blue line in Fig. 8) among the entire traces. Intuitively, we can determine that the occurrence of such an outlier is closely related to the execution times of activities $\alpha_{13}$ and $\alpha_{7}$.

The result of correlation analysis for the trace type 12 is represented in Fig. 9. The correlation matrix displays correlation relationships between activities and traces as well as relationships among activities. The activity $\alpha_{7}$ is the key steering activity to the temporal behavior of the traces because their calculated correlation coefficient is exactly 1.0. The activity $\alpha_{13}$ is another activity, considerably influencing the trace-level execution times because the correlation coefficient is 0.68.

5. Conclusion

To detect and analyze temporal outliers of business processes, we present the modeling of temporal behaviors and propose a correlation-based temporal outlier analysis method. Specifically, our aim in this study is to measure the execution times in multi-levels (trace and activity) from event logs to ultimately uncover relationships among them in terms of temporal behavior. In conclusion, we expect that our method enables us to determine, which activity is the most influential in causing the anomalous behavior of its trace.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT, Grant No. NRF-2018R1C1B5086414).

References

[1] F.M. Maggi, C.D. Francescomarino, M. Dumas, and C. Ghidini, “Predictive monitoring of business processes,” Proc. 26th International Conference on Advanced Information Systems Engineering, vol.8484, pp.457–472, 2014.
[2] A. Rogge-Solti and G. Kasneci, “Temporal anomaly detection in business processes,” Proc. 12th International Conference on Business Process Management, vol.8659, pp.234–249, 2014.
[3] M.C. Hao, D.A. Keim, U. Dayal, and J. Schneidewind, “Business process impact visualization and anomaly detection,” Information Visualization, vol.5, no.1, pp.15–27, 2006.
[4] K. Kwanghoon and C.A. Ellis, “ICN-based workflow model and its advances,” In Handbook of Research on Business Process Modeling, pp.142–171, IGI Global, 2009.
[5] Analytical Methods Committee, “Robust statistics-how not to reject outliers. Part 1. Basic concepts,” Analyst, vol.114, no.12, pp.1693–1697, 1989.

[6] 4TU, “Sepsis cases - Event log,” https://data.4tu.nl/repository/uuid:915d2bf2-7e84-49ad-a286-dc35f063a460, accessed Oct. 20, 2018.

[7] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst, “Discovering block-structured process models from event logs-a constructive approach,” Proc. 34th International Conference on Applications and Theory of Petri Nets and Concurrency, vol.7927, pp.311–329, 2013.