Automated Segmentation of User Interface Logs Using Trace Alignment Techniques (Extended Abstract)

Simone Agostinelli
Sapienza Università di Roma, Rome, Italy
agostinelli@diag.uniroma1.it

***Abstract*—Robotic Process Automation (RPA) is a fast-emerging automation technology that allows organizations to automate high volume routines. RPA tools are able to capture in dedicated User Interface (UI) logs the execution of routines previously performed by a human user on the UI of a computer system, and then emulate their enactment in place of the user by means of a software (SW) robot. The issue to automatically understand which user actions contribute to a specific routine inside the UI log is also known as *segmentation*. The proposed research investigates how to leverage trace alignment techniques in Process Mining to automatically derive the boundaries of a routine by analyzing the UI log that keeps track of its execution, thus tackling the segmentation issue.**

I. Introduction

Robotic Process Automation (RPA) uses *software robots* (or simply *SW robots*) to mimic and replicate the execution of highly routine tasks (in the following, called *routines*) performed by humans in their application's User Interface (UI). SW robots encode, by means of executable scripts, sequences of fine-grained interactions with a computer system. Commercial RPA tools allow SW robots to automate a wide range of routines in a record-and-replay fashion. The current practice for identifying the single steps of a routine is by means of interviews, walk-throughs, and detailed observation of workers conducting their daily work. A recent approach proposed by Bosco et al. [1] makes this identification less time-consuming and error-prone, as it enables to automatically extract from a UI log, which records the UI interactions during a routine enactment, those routine steps to be automated with a SW robot. While this approach is effective in case of UI logs that keep track of single routine executions, i.e., there is an exact 1:1 mapping among a recorded user action and the specific routine it belongs to, it becomes inadequate when the UI log records information about several routines whose actions are mixed in some order that reflects the particular order of their execution by the user. In addition, since the same user action may belong to different routines, the automated identification of those user actions that belong to a specific routine is far from being trivial. The challenge to automatically understand which user actions contribute to which routines inside a UI log is also known as *segmentation* [2], [3].

In this research, we investigate a technique for automatically deriving the boundaries of a routine by analyzing the UI log that keeps track of its execution, in order to cluster all user actions associated with the routine itself in well bounded *routine traces*. A routine trace represents an execution instance of a routine within a UI log. To be more precise, starting from a UI log previously recorded by a RPA tool and an *interaction model* representing the expected behaviour of a routine performed during an interaction session with the UI, we propose to leverage *trace alignment* in Process Mining [4] to automatically identify and extract the routine traces by the UI log. Such traces are finally stored in a dedicated *routine-based log*, which captures exactly all the user actions happened during many different executions of the routine, thus achieving the segmentation task.

II. Segmentation Using Trace Alignment

In this section, after providing the relevant background on trace alignment (see Section II-A), we present a first approach to tackle the segmentation issue (see Section II-B).

A. Alignment between UI Logs and Interaction Models

Trace alignment [4] is a conformance checking technique within Process Mining that is employed to replay the content of any trace of an event log against a process model represented as a Petri net, one event at a time. For each trace in the log, the technique identifies the closest corresponding trace that can be parsed by the model, i.e., an *alignment*, together with a *fitness* value, which quantifies how much the trace adheres to the process model. The fitness value can vary from 0 to 1. A fitness value equals to 1 means a perfect matching between the trace and the model.

In our context, we perform trace alignment by constructing an alignment γ of a UI log U (note that we can consider the entire content of the UI log as a single trace) and an interaction model w as a Petri net, which allows us to exactly pinpoint where deviations occur. To this aim, the events in U need to be related to transitions in the model. Building this alignment requires to relate "moves" in the log to "moves" in the model. However, it may be that some of the moves in the log cannot be mimicked by the model and vice versa. A move in log for a transition t indicates that t occurred when not allowed; a move in model for a transition t indicates that t did not occur, when, conversely, expected. Many alignments are possible for the same UI log and a Petri net. We aim at finding a complete alignment γ^{opt} of U and w with minimal number of deviations (i.e., of moves in log/model), also known in literature as *optimal alignments*. For the sake of simplicity, we are assuming here that all the deviations have the same severity. However, the severity of a deviation can be customized on a ad-hoc basis [5].

B. A First Approach to Segmenting UI Logs

The proposed approach underlying our segmentation technique consists of two methodological phases, *filtering* and *trace alignment*, to be applied in sequence. The envisioned technique takes in input a UI log U, a set of interaction models W_{set} and returns a set of routine-based logs U_{set}. For each interaction model $w \in W_{set}$ (one for each routine of interest) represented as Petri nets, the following steps are performed:

Filtering. The *filtering phase* is used to filter out noisy actions from the UI log. Specifically, for each interaction model $w \in W_{set}$, a local copy of the UI log U^w is created. Then, all user actions that appear in U^w but that can not be replayed by any transition t of w are removed from U^w. The output of this step is a *model-based filtered UI log* U^w_ϕ. Working with U^w_ϕ rather than with U^w will allow us to apply the trace alignment technique neglecting all the potential moves in log with user actions that could never be replayed by w. As a consequence, this will drastically reduce the number of alignment steps required to find optimal alignments, and at the same time optimize the overall performance. Before moving to the next step, a new routine-based log U^w_R is initialized.

Trace Alignment. The second step consists of applying the trace alignment discussed in Section II-A for any interaction model $w \in W_{set}$ and its associated model-based filtered UI log U^w_ϕ. This enables to extract from U^w_ϕ all those user actions that match a distinguishable pattern with w in the form of an optimal alignment γ^{opt}. Trace alignment allows to pinpoint the *synchronous moves* between U^w_ϕ and w. If they exist, the user actions involved in synchronous moves are extracted and stored into γ^{opt}_{sm}. Note that focusing just on synchronous moves allows us to automatically exclude all redundant user actions from the analysis. Then:

1) a trace τ_{sm} consisting of the user actions associated with the synchronous moves stored in γ^{opt}_{sm} is created;
2) a UI log U^w_{sm} containing only τ_{sm}, which is required to properly run (again) trace alignment is created;
3) a new alignment between U^w_{sm} and w with the goal to compute the fitness value is performed.

In case the fitness value is equal to 1, this means that U^w_{sm} (and, consequently, τ_{sm}) can be replayed from the start to the final marking of w, making τ_{sm} a valid routine trace of w. In such a case, τ_{sm} is stored into U^w_R and all the events associated to the synchronous moves in τ_{sm} are removed by U^w_ϕ. On the contrary, a fitness value lower than 1 indicates the presence of at least one move in the model in τ_{sm} with respect to w, i.e., τ_{sm} can not be completely replayed by w and is not a valid routine trace, meaning that we can discard it.

The above two steps can be repeated until γ^{opt}_{sm} is not empty, i.e., until there are synchronous moves in the computed alignment. At the end of the iteration, the routine-based log U^w_R is stored into U_{set}, and the the next interaction model contained in W_{set} can be analyzed. In conclusion, a number of routine-based logs equal to the number of interaction models under study are computed.

III. Discussion, Future Work and Conclusion

Our first solution to the segmentation issue is a supervised technique that leverages trace alignment to identify sequences of user actions in a UI log that belong to specific routine executions, clustering them in well bounded routine traces. Differently from event abstractions techniques [6], which map low-level *event types* to multiple high-level activities (while the *event instances*, i.e., with a specific timestamp in the log, can be coupled with a single high-level activity), segmentation techniques must enable to associate low-level event instances (corresponding to our UI actions) to multiple routines. The complete knowledge of the interaction models' structure is, of course, the main limitation of the presented technique.

As a future work, we aim at relaxing the supervised assumption in different ways: *(i)* by employing declarative rules rather than Petri nets to represent only a partial view of the routines' structure; *(ii)* by investigating *sequential pattern mining* techniques [7] to examine frequent sequences of UI actions with common data attributes; *(iii)* by analyzing *web log mining* techniques [8], which are focused on an issue similar to the one of segmentation, as the input is a set of clickstreams and the goal is to extract sessions where a user engages with a web application to fulfill a goal; *(iv)* by employing *machine learning* techniques to automatically identify routine traces without any previous knowledge of the routines' structure.

Finally, we are going to perform a robust evaluation of the proposed technique against synthetic and real-world case studies with heterogeneous UI logs. It is worth to notice that for the computation of the trace alignment, we will rely on the highly-scalable and performing planning-based alignment techniques implemented in [5], [9], which we can customize for our purposes. For this reason, our main target will be to analyze the reliability and accuracy of our technique.

References

[1] A. Bosco, A. Augusto, M. Dumas, M. La Rosa, and G. Fortino, "Discovering Automatable Routines From User Interaction Logs," in *BPM'19, Forum track*, 2019.

[2] S. Agostinelli, A. Marrella, and M. Mecella, "Research Challenges for Intelligent Robotic Process Automation," in *BPM'19 Workshops*, 2019.

[3] V. Leno, A. Polyvyanyy, M. Dumas, M. La Rosa, and F. M. Maggi, "Robotic Process Mining: Vision and Challenges," *BISE*, 2020.

[4] A. Adriansyah, N. Sidorova, and B. F. van Dongen, "Cost-Based Fitness in Conformance Checking," in *ACSD'11*. IEEE, 2011.

[5] M. de Leoni and A. Marrella, "Aligning Real Process Executions and Prescriptive Process Models through Automated Planning," *Expert System with Application*, vol. 82, pp. 162–183, 2017.

[6] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. van der Aalst, and P. J. Toussaint, "Guided process discovery – a pattern-based approach," *Inf. Syst.*, vol. 76, 2018.

[7] G. Dong, *Sequence Data Mining*. Springer-Verlag, 2009.

[8] *Web Usage Mining*. Springer Berlin Heidelberg, 2007, pp. 449–483.

[9] G. De Giacomo, F. M. Maggi, A. Marrella, and F. Patrizi, "On the Disruptive Effectiveness of Automated Planning for LTLf-Based Trace Alignment," in *AAAI'17*, 2017.