A Human-in-the-loop Approach to Support the Segments Compliance Analysis

Simone Agostinelli, Giacomo Acitelli, Michela Capece, and Massimo Mecella

Sapienza Universitá di Roma, Rome, Italy
{acitelli.1643776,capece.1694700}@studenti.uniroma1.it
{agostinelli,mecella}@diag.uniroma1.it

Abstract. Robotic Process Automation (RPA) is an emerging automation technology in the field of Business Process Management (BPM) that creates software (SW) robots to partially or fully automate rule-based and repetitive tasks (a.k.a. routines) previously performed by human users in their applications' user interfaces (UIs). Nowadays, successful usage of RPA requires strong support by skilled human experts, from the discovery of the routines to be automated (i.e., the so-called segmentation issue of UI logs) to the development of the executable scripts required to enact SW robots. In this paper, we present a human-in-the-loop approach to filter out the routine behaviors (a.k.a. routine segments) not allowed (i.e., wrongly discovered from the UI log) by any real-world routine under analysis, thus supporting human experts in the identification of valid routine segments. We have also measured to which extent the human-in-the-loop strategy satisfies three relevant non-functional requirements, namely effectiveness, robustness and usability.

Keywords: Robotic Process Automation · Segmentation of UI logs · Declarative Constraints

1 Introduction

Robotic Process Automation (RPA) is an emerging automation technology in the Business Process Management (BPM) domain [17] that creates software (SW) robots to partially or fully automate rule-based and repetitive tasks (or simply *routines*) performed by human users in their applications' user interfaces (UIs) [1] of their computer systems.

To date, the identification of the routine steps to robotize from a UI log require the support of skilled human experts, which need to [16]: *(i)* preliminary observe how routines are executed on the UI of the involved SW applications (by means of walkthroughs, etc.), *(ii)* convert such observations in explicit flowchart diagrams, which are specified to show all the potential behaviours of the routines of interest, and *(iii)* finally implement the SW robots that automate the routines enactment on a target computer system. However, the current practice is time-consuming and error-prone, as it strongly relies on the ability of human experts to correctly interpret the routines to automate.

For tackling this challenge, in their Robotic Process Mining framework [20], Leno et al. propose to exploit the User Interface (UI) logs recorded by RPA tools to automatically discover the candidate routines that can be later automated with SW robots. UI logs are sequential data of user actions performed on the UI of a computer system during many routines' executions. Typical user actions are: opening a file, selecting/copying a field in a form or a cell in a spreadsheet, read and write from/to databases, open emails and attachments, etc.

Nowadays, when considering state-of-the-art RPA technology, it is evident that the RPA tools available in the market are not able to learn how to automate routines by only interpreting the user actions stored into UI logs [4]. The main trouble is that in a UI log there is not an exact 1:1 mapping among a recorded user action and the specific routine segment it belongs to. *Routine segments* describe the different behaviours of the routine(s) under analysis, in terms of repeated patterns of performed user actions. In fact, the UI log usually records information about several routines whose actions are mixed in some order that reflects the particular order of their execution by the user. The issue to automatically understand which user actions contribute to a particular routine segment inside a UI log and cluster them into well-bounded *routine traces* (i.e., complete execution instances of a routine) is known as *segmentation* [4,20].

The majority of state-of-the-art segmentation approaches are able to properly extract routine segments from unsegmented UI logs when the routine executions are not interleaved from each others. Only few works are able to partially untangle unsegmented UI logs consisting of many interleaved routines executions, but with the assumption that any routine provides its own, separate universe of user actions. This is a relevant limitation, since it is quite common that real-world routines may share the same user actions (e.g., copy and paste data across cells of a spreadsheet) to achieve their objectives. The limitations mentioned above are addressed in [3], where we proposed a new approach to the discovery of routine traces from unsegmented UI logs, that is able to segment a UI log that records in an interleaved fashion many different routines with shared user actions but not the routine executions, thus losing in accuracy when there is the presence of interleaving executions of the same routine.

Specifically, the technique presented in [3] may discover routine segments that represent not allowed routine behaviours. This happens because a UI log combines the execution of several routines that are usually interleaved from each others. In addition, in case of routines that make use of the same kinds of user actions to achieve their goals, it may happen that new patterns of repeated user actions, which represent potential not allowed routine segments, are rather detected as valid ones within the UI log. In this paper, starting from [3], we present: (1) a human-in-the-loop approach together with its implemented tool called SCAN[1] (Segments Compliance ANalysis), that allows users to filter out those routines' segments not compliant with any real-world routine behaviours (thus supporting human experts in performing the segmentation task), and (2) the results of the multi-step evaluation conducted on SCAN.

[1] The tool can be downloaded at: `https://github.com/bpm-diag/SCAN`

The rest of the paper is organized as follows. Section 2 introduces a running example that will be used to explain our approach, then it discusses literature works on the segmentation of UI logs. Section 3 presents the required steps to enact the human-in-the-loop strategy through SCAN, instantiating it over the RPA use case of Section 2. Section 4 measures the impact of the human-in-the-loop strategy to filter out the wrongly discovered routine segments. Specifically, we present the results of SCAN to investigate to which extent it satisfies three relevant non-functional requirements, namely *effectiveness*, *robustness* and *usability*. The target is to understand if SCAN can potentially complement the traditional solutions provided by open-source Process Mining tools for helping users to perform the segmentation task in RPA. Finally, Section 5 draws conclusions and traces future works.

2 Background

2.1 Running Example

In this section, we describe an RPA use case inspired by a real-life scenario at the Department of Computer, Control and Management Engineering (DIAG) of Sapienza Università di Roma, which has already proved to be effective in ours previous works [3,5]. The scenario concerns the filling of the travel authorization request form made by professors, researchers and PhD students of DIAG for travel requiring prior approval. The request applicant must fill a well-structured Excel spreadsheet (cf. Fig. 1(a)) providing some personal information, such as her/his bio-data and the email address, together with further information related to the travel, including the destination, the starting/ending date/time, the means of transport to be used, the travel purpose, and the envisioned amount of travel expenses, associated with the possibility to request an anticipation of the expenses already incurred. When ready, the spreadsheet is sent via email to an employee of the Administration Office of DIAG, which is in charge of approving and elaborating the request. Concretely, for each row in the spreadsheet, the employee manually copies every cell in that row and pastes that into the corresponding text field in a dedicated Google form (cf. Figure 1(b)), accessible just by the Administration staff. Once the data transfer for a given travel authorization request has been completed, the employee presses the "Submit" button to submit the data into an internal database.

We denote this routine procedure as $R_{example}$. In particular, the path of user actions performed by the Administration employee in the UI to properly enact $R_{example}$ is as follows:

- loginMail, to access the client email;
- accessMail, to access the specific email with the travel request;
- downloadAttachment, to download the Excel file including the travel request;
- openWorkbook, to open the Excel spreadsheet;
- openGoogleForm, to access the Google Form to be filled;
- getCell, to select the cell in the i-th row of the Excel spreadsheet;

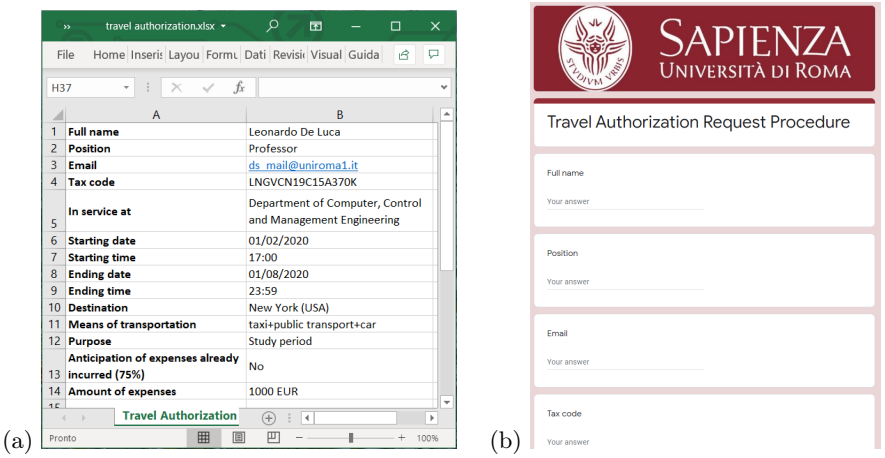

Fig. 1. UIs involved in the running example

- copy, to copy the content of the selected cell;
- clickTextField, to select the specific text field of the Google form where the content of the cell should be pasted;
- paste, to paste the content of the cell into the corresponding text field of the Google form;
- formSubmit, to press the button to finally submit the Google form to the internal database.

Note that, the user actions openWorkbook and openGoogleForm can be performed in any order. Moreover, the sequence of actions ⟨getCell, copy, clickTextField, paste⟩ will be repeated for any travel information to be moved from the Excel spreadsheet to the Google form.

For example, a valid routine segment of $R_{example}$ is ⟨loginMail, accessMail, downloadAttachment, openWorkbook, openGoogleForm, getCell, copy, clickTextField, paste, formSubmit⟩. Valid routine segments are also those ones where: *(i)* loginMail is skipped (if the user is already logged in the client email); *(ii)* the pair of actions ⟨openWorkbook, openGoogleForm⟩ is performed in reverse order; *(iii)* the sequence of actions ⟨getCell, copy, clickTextField, paste⟩ is executed several time before submitting the Google form.

2.2 Segmentation in RPA

Given a UI log that consists of events including user actions with the same granularity[2] and potentially belonging to different routines, in the RPA domain *segmentation* is the task of clustering parts of the log together which belong to

[2] The UI logs created by generic action loggers usually consist of low-level events associated one-by-one to a recorded user action on the UI (e.g., mouse clicks, etc.).

the same routine. In a nutshell, the challenge is to automatically understand which user actions contribute to which routines and organize such user actions in well-bounded routine traces [4,20].

In [6] we identified three main forms of UI logs and their segmentation variants, which can be categorized according to the fact that: *(i)* any user action in the log exclusively belongs to a specific routine; *(ii)* the log records the execution of many routines that do not have any user action in common; *(iii)* the log records the execution of many routines, with the possibility that some performed user actions are shared by many routines at the same time. In the following, we analyze literature works in terms of supported segmentation variants.

Concerning RPA-related techniques, Bosco et al. [10] provide a method that exploits rule mining and data transformation techniques, able to discover routines that are fully deterministic and thus amenable for automation directly from UI logs. The method combines a technique for compressing a set of sequences of user actions into an acyclic automaton using rule mining techniques and data transformations. This approach is effective in the case of UI logs that keep track of well-bounded routine executions and becomes inadequate when the UI log records information about several routines whose actions are potentially interleaved. In this direction, Leno et al. [19] propose a technique to identify execution traces of a specific routine relying on the automated synthesis of a control-flow graph, describing the observed directly-follow relations between the user actions. The technique in [19] loses in accuracy in the presence of recurrent noise and interleaved routine executions while it is not able to handle UI logs that record in an interleaved fashion shared user actions of many different routines.

Even if more focused on traditional business processes in BPM rather than on RPA routines, Bayomie et al. [9] address the problem of correlating uncorrelated event logs in process mining in which they assume the model of the routine is known. Since event logs allow to store traces of one process model only, this technique is able to handle logs recording user actions belonging to a specific routine. In the field of process discovery, Măruşter et al. [24] propose an empirical method for inducing rule sets from event logs containing the execution of one process only. Therefore, as in [9], this method is able to partially achieve the first case, thus making the technique ineffective in the presence of interleaved and shared user actions. A more robust approach, developed by Fazzinga et al. [12], employs predefined behavioural models to establish which process activities belong to which process model. The technique works well when there are no interleaved user actions belonging to one or more routines since it cannot discriminate which event instance (but just the event type) belongs to which process model. This makes [12] effective to partially tackle all three cases. Closely related to [12], there is the work of Liu [21]. The author proposes a probabilistic approach to learn workflow models from interleaved event logs, dealing with noises in the log data. Since each workflow is assigned with a disjoint set of operations, it means the proposed approach is able to partially achieve first two cases (the approach can lose accuracy in assigning operations to workflows).

Differently from the previous works, Time-Aware Partitioning (TAP) techniques cut event logs based on the temporal distance between two events [28,18]. The main limitation of TAP approaches is that they rely only on the time gap between events without considering any process/routine context. For this reason, such techniques cannot handle neither interleaved user actions of different routine executions nor interleaved user actions of different routines.

There exist other approaches whose target is not to exactly resolve the segmentation issue. Many research works exist that analyze UI logs at different abstraction levels, which can be potentially valuable for realizing segmentation techniques. With the term *"abstraction"* we mean that groups of user actions to be interpreted as executions of high-level activities. Baier et al. [8] propose a method to find a global one-to-one mapping between the user actions that appear in the UI log and the high-level activities of a given interaction model. This method leverages constraint-satisfaction techniques to reduce the set of candidate mappings. Similarly, Ferreira et al. [13], starting from a state-machine model describing the routine of interest in terms of high-level activities, employ heuristic techniques to find a mapping from a "micro-sequence" of user actions to the "macro-sequence" of activities in the state-machine model. Finally, Mannhardt et al. [23] present a technique that maps 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). However, segmentation techniques in RPA must enable to associate low-level event instances (corresponding to user actions) to multiple routines, making abstractions techniques ineffective to tackle all those cases where is the presence of interleaving user actions of the same (or different) routine(s).

The analysis of the related work has pointed out that the majority of literature approaches are able to properly extract routine traces from unsegmented UI logs when the routine executions are not interleaved from each others, which is far from being a realistic assumption. Only a few works [12,5,19,21] have demonstrated the full or partial ability to untangle unsegmented UI logs consisting of many interleaved routine executions, but with any routine providing its own, separate universe of user actions. However, we did not find any literature work able to properly deal with user actions potentially shared by many routine executions in the UI log. This is a relevant limitation since it is quite common that a user interaction with the UI corresponds to the executions of many routine steps at once. Moreover, it is worth noticing the majority of the literature works rely on the so-called *supervised* assumption, which consists of some a priori knowledge of the structure of routines. Of course, this knowledge may ease the task of segmenting a UI log. But, as a side effect, it may strongly constrain the discovery of routine traces only to the "paths" allowed by the routines' structure, thus neglecting that some valid infrequent routine variants may exist in the UI log.

Finally, we want to underline that process discovery techniques [7] can also play a relevant role in tackling the segmentation issue, as demonstrated by some literature works [21,12,9]. However, the problem is that most discovery techniques work with event logs containing behaviours related to the execution of

a single process model only. And, more importantly, event logs are already segmented into traces, i.e., with clear starting and ending points that delimitate any recorded process execution. Conversely, a UI log consists of a long sequence of user actions belonging to different routines without any clear starting/ending point. Thus, a UI log is more similar to a unique trace consisting of thousands of fine-grained user actions. With a UI log as input, the application of traditional discovery algorithms seems unsuited to discover routine traces and associate them to some routine models, even if more research is needed in this area.

3 Segments Compliance Analysis

The main limitations of state-of-the-art segmentation techniques are tackled in [3]. Here, we have presented a new approach to the automated segmentation of UI logs which is able to extract routine traces from unsegmented UI logs that record in an interleaved fashion many different routines. Specifically, in [3] when routine segments have been discovered from a UI log, there exists the possibility that many of them represent not allowed routine behaviours. This happens because a UI log combines the execution of several routines that are usually interleaved from each others. In addition, in case of routines that make use of the same kinds of user actions to achieve their goals, it may happen that new patterns of repeated user actions, which represent potential not allowed routine segments, are rather detected as valid ones within the UI log.

Towards this direction, we realized a stand-alone web application called SCAN[3] (*Segments Compliance ANalysis*), which allows to support human experts in performing the segmentation task. The tool enables to visualize the declarative constraints (i.e., the temporally extended relations between user actions) that must be satisfied throughout the discovered routine segments from the UI log. The constraints are represented using Declare, a well-known declarative process modeling language introduced in [25]. This knowledge allows human experts to identify and remove those constraints that should not be compliant with any real-world routine behaviour. Detecting and removing these constraints means to filter out all the not allowed (i.e., wrongly discovered) routine segments from the UI log, as shown in Figure 2. Declare constraints can be divided into four main groups: existence, relation, mutual and negative constraints. We notice that the use of declarative notations has been already demonstrated as an effective tool to visually support expert users in the analysis of event logs [26].

For instance, if we consider the following valid routine segment of $R_{example}$ (cf. Section 2.1): ⟨loginMail, accessMail, downloadAttachment, openWorkbook, open GoogleForm, getCell, copy, clickTextField, paste, formSubmit⟩, then these Declare constraints must hold:

- *Start*(loginMail)
- *Precedence*(getCell,copy), *Precedence*(clickTextField,paste)
- *End*(formSubmit)

[3] SCAN can be downloaded at: `https://github.com/bpm-diag/SCAN`

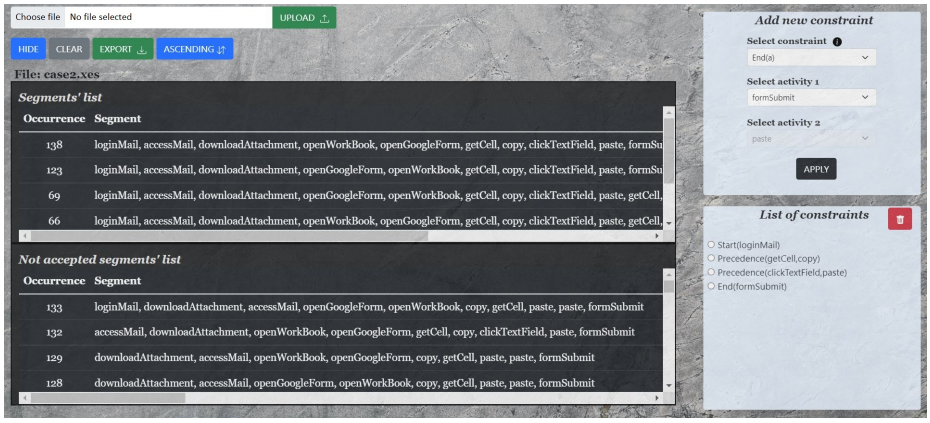

Fig. 2. GUI of SCAN

An expert user that is aware of the behaviour of the real-world routines under analysis can immediately understand that those segments not satisfying the above Declare constraints should be filtered out. For this reason, the above Declare constraints can be considered representative of $R_{example}$. As a consequence, all the discovered segments for which one of the above Declare constraints does not hold can be immediately discarded.

We point out that the iterative analysis of the Declare constraints associated to the discovered segments will support the human experts to easily detect and filter out those segments that must not be later emulated by SW robots.

4 Evaluation

In this section, we present the results of a multi-step evaluation performed on SCAN to investigate the extent to which our approach satisfies three relevant non-functional requirements, namely *effectiveness*, *robustness*, and *usability*. The target is to understand if SCAN can potentially complement the traditional solutions provided by open-source Process Mining tools for performing the segmentation task in RPA.

4.1 Evaluating the Effectiveness of SCAN

An approach that simplifies the segmentation task in RPA, and in particular the inspection of routine segments required to filter out the not allowed ones, can be considered as a relevant artefact to investigate. Consequently, the research question (**RQ**) we aim to investigate is the following one: "*What is the effectiveness of employing an approach that semi-automatically filters out the not allowed routine segments, thus neglecting the (manual) identification stage of the not allowed real-word routine behaviour, through declarative constraints?*".

In order to address **RQ** we enacted a controlled experiment involving a sample of 18 Master students of the course of Process Management and Mining (PMM) held at Sapienza University of Rome, to investigate the effectiveness of employing SCAN to perform the segmentation task when compared to Disco[4]. Specifically, we selected Disco as target Process Mining tool since it provides user-friendly functionalities, integrated with filtering facilities that allows to filter out the not allowed routine segments as stored into event logs.

The user study was conducted as follows. Two case studies of increasing complexity were submitted to two different user groups of PMM students. The provided case studies are inspired by the one presented in Section 2.1 and we refer to them as Case Study #1 and Case Study #2. A first group of 9 PMM students were instructed to perform the case studies #1 and #2 exclusively with Disco. We denote with p_1 this first group of users. In parallel, a second group of 9 PMM students received the same instructions of group p_1 but they are asked to use SCAN rather than Disco. We denote with p_2 this second group of users. It is worth noticing that all the PMM students involved in the user study can be considered expert users in business process modelling and automation.

To assess the effectiveness of SCAN in filtering out the not allowed segments, we investigated the following experimental hypothesis H_1: *Employing SCAN, thus neglecting the manual identification stage of the not allowed real-word routine behaviour through declarative constraints, is more effective than employing traditional approaches (e.g. Disco) that require to manually identify and filter out the not allowed routine segments.* To validate H_1, a *between-subject approach* was used, i.e., each user in p_1 (p_2, respectively) was assigned to a different experimental condition, related to the exclusive use of SCAN (c_1) or Disco (c_2) to perform the required steps for accomplishing both case studies. Any user in p_1 was preliminarily instructed about the functionalities of SCAN through a short training session, while the users in p_2 already know how to use Disco.

We evaluated the validity of H_1 by asking any user expert that completed the user study the following three questions:

- Q_1: *The segment's filtering process required to filter out the not allowed routine segments is a complex task. Do you agree?*
- Q_2: *The inspection of the routine segments is a complex task. Do you agree?*
- Q_3: *SCAN (Disco, respectively) makes the segmentation task feasible. Do you agree?*

Questions are rated with a 5-point Likert scale ranging from 1 ("strongly disagree") to 5 ("strongly agree"). To validate Q_1, Q_2 and Q_3 we performed a comparison of the rates obtained from the questionnaire, respectively in the cases of c_1 and c_2. Specifically, for each question, we employed a *2-Sample t-test* with a 95% confidence level to determine whether the means between the two distinct populations (i.e., p_1 and p_2) involved in c_1 and c_2 differ. We measured the level of statistical significance by analyzing the resulting p-value. We remind that a $p - value \leq 0.05$ is considered to be statistically significant, while a

[4] `https://fluxicon.com/disco/`

$p - value \leq 0.01$ indicates that there is substantial evidence in favour of the experimental hypothesis. The results of the analysis are summarized in Table 1 that shows the values sorted in descending order, assigned to the responses of each user.

Table 1. Effectiveness of SCAN: p-values associated to each question

Q_1		Q_2		Q_3	
DISCO	*SCAN*	*DISCO*	*SCAN*	*DISCO*	*SCAN*
5	4	5	4	4	5
4	4	5	3	4	5
4	3	5	3	3	5
4	2	5	3	3	5
4	2	4	2	2	5
3	2	4	2	2	4
3	2	4	1	1	4
2	2	3	1	1	4
1	2	2	1	1	4
p-value: 0.1443957		p-value: 0.0018155		p-value: 0.0005373	

It appears evident that the experimental hypothesis H_1 is statistically supported by the results obtained for Q_2 and Q_3, while it is rejected for Q_1. Concerning Q_1, *it seems that the segment's filtering process was relatively easier in SCAN with respect to Disco.* Still, there is no statistical difference among the two distinct populations since for Q_1, the p-value obtained is 0.1443957, which is greater than 0.05, and this means that hypothesis H_1 is rejected on Q_1. *On the other hand, the inspection of routine segments in Disco seems to be more complex than SCAN* since, for Q_2, the p-value obtained is 0.0018155, which is less than 0.05, and this means that the hypothesis H_1 is accepted on Q_2. Finally, for Q_3, we got a p-value equal to 0.0005373, which is less than 0.05, and this means the hypothesis H_1 is accepted on Q_3. In particular, this value is less than 0.01, meaning that there is a substantial difference between the means of the two distinct populations. This is reflected in higher values associated with SCAN and lower values associated with Disco, *thus making the segmentation task more feasible in SCAN with respect to Disco.* Therefore, H_1 can be considered partially accepted since it is validated for both Q_2 and Q_3 but rejected for Q_1, where there is no statistical evidence that the use of SCAN is more effective than traditional process mining solutions (e.g., Disco) in the process of segment's filtering.

4.2 Assessing the Robustness of SCAN

To investigate the robustness of SCAN to the achievement of user tasks specified in both Case Study #1 and Case Study #2, we collected the event logs resulting as an output of the user study, and then we compared them with the ground truth event logs (i.e., we computed a priori the event logs as results of the case

studies). Precisely, the *robustness* is measured as the ratio between the number of logs compliant with the ground truth logs and the total number of logs, both for p_1 (i.e., SCAN) and p_2 (i.e., Disco) grouped by Case (i.e., Case Study #1 and Case Study #2).

In the following, we will show the results obtained both for Case Study #1 and for Case Study #2. Note that both the populations p_1 and p_2 first executed Case Study #1 in a limited time of 10 minutes and then Case Study #2, considered more complex, in 20 minutes.

- **Case Study #1**. Both p_1 and p_2 had 10 minutes to read the assigned track and run the task either on Disco (i.e., p_2) or SCAN (i.e., p_1) respectively. For p_2, it is important to remember that users already know how to use the tool. The results obtained in this case is that 8 people out of 9 have executed the task arriving at the right event log, while 1 has obtained a wrong result. Thus, the robustness in case of p_2 is as follows $Robustness_{p_2} = \frac{8}{9} = 0.88$.
  On the other hand, for p_1, we remind the reader that the users experienced SCAN for the first time during this experiments session. In this case, the number of users who achieved the right result is 6 out of 9, while 3 have computed a wrong event log. Therefore, the robustness in case of p_1 is $Robustness_{p_1} = \frac{6}{9} = 0.66$.
- **Case Study #2**. This case was executed immediately after the first one. The time allowed for achieving the task was 20 minutes due to the major complexity with respect to the previous one. For the class of users belonging to p_2, the result obtained was that 4 out of 9 people have computed the right result while 5 the wrong one. It follows that the robustness in case of p_2 is $Robustness_{p_2} = \frac{4}{9} = 0.44$.
  On the contrary, users assigned to p_1 performs much better. Indeed, 7 users among 9 computed the right result, while 2 the wrong one. As a consequence, the correctness for the users that used SCAN is $Robustness_{p_1} = \frac{7}{9} = 0.77$.

If we make a comparison between the degree of robustness for both SCAN and Disco in each case study, it can be stated that:

- *For Case Study #1, better results are achieved with Disco.* This is because the original log contains solely 8 routine segments, and among these only 4 were correct. For this reason, they were easily identifiable and therefore easy to be manually filtered. Regarding SCAN, we can say that since this was the first time the users experienced the tool, it is possible that the limited time of 10 minutes was not enough for completing the task. In addition, it is also possible that users had not yet settled into using SCAN even if they had been instructed during the short training session.
- *On the other hand, for Case Study #2, better results are achieved with SCAN.* Since the original log presents more than 80 routine segments, the manual identification stage of the wrong routine segments makes the filtering steps even more challenging with Disco (that required the users to filter the wrong routine segments one by one) rather than with SCAN. Indeed, through

SCAN, it is possible to apply a limited number of declarative constraints to filter out a large number of wrong routine segments, thus neglecting the manual identification stage of Disco. In addition, the learning effective plays an essential role in the achievement of good results since users trained themselves while completing the task outlined in Case Study #1. This learning experience is thus reflected in the accomplishment of Case Study #2.

4.3 Quantifying the Usability of the UI of SCAN

We investigated the degree of *usability* of the UI developed for SCAN through the administration of the SUS (Software Usability Scale) questionnaire (which is one of the most widely used methodologies to measure the users' perception of the usability of a tool [11]) to the 9 PMM students that were involved in the experimental condition c_1, i.e., that used SCAN. The questionnaire consists of 10 statements, adapted to SCAN and, evaluated with a Likert scale that ranges from 1 ("strongly disagree") to 5 ("strongly agree"):

q1) I think that I would like to use SCAN frequently.
q2) I found SCAN unnecessary complex.
q3) I thought SCAN was easy to use.
q4) I think that I would need the support of a technical person to be able to use SCAN.
q5) I found the various functions in SCAN well integrated.
q6) I thought there was too much inconsistency in SCAN.
q7) I would imagine that most people would learn to use SCAN very quickly.
q8) I found SCAN very awkward to use.
q9) I felt very confident using SCAN.
q10) I needed to learn a lot of things before I could get going with SCAN.

Table 2. Computation of the SUS overall score

participant	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	SUS Score	Average
p1	5	1	5	1	5	1	5	1	5	1	100	82,5
p2	5	2	4	1	4	2	5	2	3	3	77,5	
p3	5	1	4	1	4	2	2	1	4	2	80	
p4	4	3	4	3	3	2	4	2	4	2	67,5	
p5	4	1	4	3	4	2	5	1	4	3	77,5	
p6	4	2	5	2	4	2	5	1	4	2	82,5	
p7	5	4	5	2	5	1	5	4	5	1	82,5	
p8	4	2	5	1	4	2	5	2	5	2	85	
p9	5	1	5	1	4	2	4	2	5	1	90	

At the end of the questionnaire, an overall score is assigned to the questionnaire. The score can be compared with several benchmarks presented in the

research literature to determine the degree of usability of the tool being evaluated. In our test, we made use of the benchmark given in [27], which associates to each range of the SUS score a percentile ranking varying from 0 to 100, indicating how well it compares to other 5,000 SUS observations performed in the literature. The collection of the ranks associated with any statement of the SUS is reported in Table 2, calculated following the steps discussed in [27].

Since the average SUS score obtained by SCAN was 82.5, according to the selected benchmark [27], the usability of the tool corresponds to a rank of A, which indicates a degree of usability almost excellent.

The result shows that the UI implemented has been comprehensive and straightforward since the first use of the tool. And also that the use of the tool has been found effective and performing in achieving the required tasks.

5 Conclusion

RPA recently gained a lot of attention in the BPM domain [1]. Since RPA operates at the UI level, rather than at the system level, it allows applying automation without any changes in the underlying information system. However, the current generation of RPA tools is driven by predefined rules and manual configurations made by expert users rather than by automated techniques [22], preventing widespread adoption of these tools in the BPM domain.

Still, to date, a great deal of time is required to identify the routines for automation and manually program the SW robots. Even if RPA tools are able to automate a wide range of routines, they cannot determine which routines should be automated in the first place. Indeed, in the early stages of the RPA life-cycle it is required to: (1) identify the candidate routines to automate through interviews and detailed observation of workers conducting their daily work, (2) record the interactions that take place during the routines' enactment on the UI of software applications into dedicated UI logs, and (3) manually specify their conceptual and technical structure (often in form of flowchart diagrams) for identifying the behaviour of SW robots. Towards this direction, the presented work tries to improve the process of segments identification (cf. step 1) performed by skilled human experts, throughout the development of a human-in-the-loop approach that support human experts in visualizing and filtering out all those segments discovered from a UI log not satisfying specific declarative constraints. We implemented our approach as a stand-alone web application called SCAN which has been evaluated through the measurement of three non-functional requirements, namely, *effectiveness*, *robustness* and *usability*.

The presented approach can be leveraged by segmentation techniques that are able to discover from scratch the structure of the routines under analysis that were previously captured in a UI log, thus increasing the quality of discovered routine segments. However, the main limitation of our approach relies on the involvement of human experts when the automated discovery of the routine segments is completed as required by the approach itself, given the possibility of complementing the unsupervised assumption with the experts' knowledge.

For this reason, we think that an important step towards the development of a more complete and unsupervised technique to the segmentation of UI logs is to shift from current semi-supervised learning approaches to completely unsupervised ones [15,2,14].

Acknowledgements This work has been supported by the H2020 project DataCloud (grant ID 101016835), the Sapienza grant BPbots, by the Italian projects Social Museum and Smart Tourism (CTN01_00034_23154) and RoMA - Resilience of Metropolitan Areas (SCN_00064).

References

1. van der Aalst, W.M.P., Bichler, M., Heinzl, A.: Robotic Process Automation. Bus. Inf. Syst. Eng. **60**(4), 269–272 (2018). https://doi.org/10.1007/s12599-018-0542-4
2. van der Aalst, W.M.P., Bose, R.J.C.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Int. Conf. on Business Process Management. pp. 159–175. Springer (2009). https://doi.org/10.1007/978-3-642-03848-8_12
3. Agostinelli, S., Leotta, F., Marrella, A.: Interactive Segmentation of User Interface Logs. In: 19th Int. Conf. on Service-Oriented Computing , ICSOC 2021. vol. 13121, pp. 65–80 (2021). https://doi.org/10.1007/978-3-030-91431-8_5
4. Agostinelli, S., Marrella, A., Mecella, M.: Research Challenges for Intelligent Robotic Process Automation. In: Business Process Management Workshops - BPM 2019 Int. Workshops, Vienna, Austria, September 1-6, 2019. pp. 12–18 (2019). https://doi.org/10.1007/978-3-030-37453-2_2
5. Agostinelli, S., Marrella, A., Mecella, M.: Automated Segmentation of User Interface Logs. In: RPA. Management, Technology, Applications. De Gruyter (2021). https://doi.org/10.1109/EDOC.2019.00026
6. Agostinelli, S., Marrella, A., Mecella, M.: Exploring the Challenge of Automated Segmentation in Robotic Process Automation. In: 15th Int. Conf. on Research Challenges in Information Science, RCIS'21 (2021). https://doi.org/10.1007/978-3-030-75018-3_3
7. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019). https://doi.org/10.1109/TKDE.2018.2841877
8. Baier, T., Rogge-Solti, A., Mendling, J., Weske, M.: Matching of Events and Activities: An Approach Based on Behavioral Constraint Satisfaction. In: ACM Symp. on Appl. Comp. pp. 1225–1230 (2015). https://doi.org/10.1145/2695664.2699491
9. Bayomie, D., Ciccio, C.D., La Rosa, M., Mendling, J.: A Probabilistic Approach to Event-Case Correlation for Process Mining. In: 38th Int. Conf. on Conceptual Modeling (ER'19). pp. 136–152 (2019). https://doi.org/10.1007/978-3-030-33223-5_12
10. Bosco, A., Augusto, A., Dumas, M., La Rosa, M., Fortino, G.: Discovering Automatable Routines From User Interaction Logs. In: Int. Conf. on Business Process Management (BPM'19), Forum track, Vienna, Austria. pp. 144–162. Springer (2019). https://doi.org/10.1007/978-3-030-26643-1_9
11. Brooke, J.: SUS: a Retrospective. Journal of Usability Studies **8**(2), 29–40 (2013)

12. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: Efficiently Interpreting Traces of Low Level Events in Business Process Logs. Information Systems **73**, 1–24 (2018). https://doi.org/10.1016/j.is.2017.11.001
13. Ferreira, D.R., Szimanski, F., Ralha, C.G.: Improving Process Models by Mining Mappings of Low-Level Events to High-Level Activities. Intelligent Information Systems **43**(2), 379–407 (2014). https://doi.org/10.1007/s10844-014-0327-2
14. Folino, F., Guarascio, M., Pontieri, L.: Mining Multi-variant Process Models from Low-Level Logs. In: Int. Conf. on Business Information Systems. pp. 165–177. Springer (2015). https://doi.org/10.1007/978-3-319-19027-3_14
15. Günther, C.W., Rozinat, A., van Der Aalst, W.M.: Activity Mining by Global Trace Segmentation. In: Int. Conf. on Business Process Management. pp. 128–139. Springer (2009). https://doi.org/10.1007/978-3-642-12186-9_13
16. Jimenez-Ramirez, A., Reijers, H.A., Barba, I., Del Valle, C.: A Method to Improve the Early Stages of the Robotic Process Automation Lifecycle. In: 31st Int. Conf. on Advanced Information Systems Engineering (CAiSE'19). pp. 446–461 (2019). https://doi.org/10.1007/978-3-030-21290-2_28
17. Kirchmer, M.: Robotic Process Automation-Pragmatic Solution or Dangerous Illusion. BTOES Insights, June'17 (2017)
18. Kumar, A., Salo, J., Li, H.: Stages of User Engagement on Social Commerce Platforms: Analysis with the Navigational Clickstream Data. Int. J. El. C. **23**(2) (2019). https://doi.org/10.1080/10864415.2018.1564550
19. Leno, V., Augusto, A., Dumas, M., La Rosa, M., Maggi, F.M., Polyvyanyy, A.: Identifying Candidate Routines for Robotic Process Automation from Unsegmented UI Logs. In: 2nd Int. Conf. on Process Mining. pp. 153–160 (2020). https://doi.org/10.1109/ICPM49681.2020.00031
20. Leno, V., Polyvyanyy, A., Dumas, M., La Rosa, M., Maggi, F.M.: Robotic Process Mining: Vision and Challenges. Business & Information Systems Engineering (2020). https://doi.org/10.1007/s12599-020-00641-4
21. Liu, X.: Unraveling and Learning Workflow Models from Interleaved Event Logs. In: 2014 IEEE Int. Conf. on Web Services. pp. 193–200 (2014). https://doi.org/10.1109/ICWS.2014.38
22. Lohr, S.: The Beginning of a Wave: A.I. Tiptoes Into the Workplace. https://www.nytimes.com/2018/08/05/technology/workplace-ai.html/ (2018)
23. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M., Toussaint, P.J.: Guided Process Discovery – A Pattern-Based Approach. Inf. Syst. **76**, 1–18 (2018). https://doi.org/10.1016/j.is.2018.01.009
24. Măruşter, L., Weijters, A.T., Van Der Aalst, W.M., Van Den Bosch, A.: A Rule-Based Approach for Process Discovery: Dealing with Noise and Imbalance in Process Logs. Data Mining and Knowledge Discovery **13**(1), 67–87 (2006). https://doi.org/10.1007/s10618-005-0029-z
25. Pesic, M., Schonenberg, H., van Der Aalst, W.M.: Declarative Workflows: Balancing between Flexibility and Support. Comp. Sc.-Res. and Dev. **23**(2) (2009). https://doi.org/10.1007/s00450-009-0057-9
26. Rovani, M., Maggi, F.M., de Leoni, M., van der Aalst, W.M.: Declarative Process Mining in Healthcare. Expert Systems with Applications **42**(23) (2015)
27. Sauro, J., Lewis, J.R.: Quantifying the User Experience: Practical Statistics for User Research. Morgan Kaufmann (2016). https://doi.org/10.1145/2413038.2413056
28. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. SIGKDD Exp. **1**(2) (2000). https://doi.org/10.1145/846183.846188