Exploring the Challenge of Automated Segmentation in Robotic Process Automation

Simone Agostinelli, Andrea Marrella, and Massimo Mecella

Sapienza Universitá di Roma, Rome, Italy
{agostinelli,marrella,mecella}@diag.uniroma1.it

Abstract. Robotic Process Automation (RPA) is an emerging technology that allows organizations to automate intensive repetitive tasks (or simply *routines*) previously performed by a human user on the User Interface (UI) of web or desktop applications. RPA tools are able to capture in dedicated UI logs the execution of several routines and then emulate their enactment in place of the user by means of a software (SW) robot. A UI log can record information about many routines, whose actions are mixed in some order that reflects the particular order of their execution by the user, making their automated identification far from being trivial. The issue to automatically understand which user actions contribute to a specific routine inside the UI log is also known as *segmentation*. In this paper, we leverage a concrete use case to explore the issue of segmentation of UI logs, identifying all its potential variants and presenting an up-to-date overview that discusses to what extent such variants are supported by existing literature approaches. Moreover, we offer points of reference for future research based on the findings of this paper.

1 Introduction

Robotic Process Automation (RPA) is an automation technology in the field of Business Process Management (BPM) that creates *software (SW) robots* to replicate rule-based and repetitive tasks (or simply *routines*) performed by human users in their applications' user interfaces (UIs). A typical routine that can be automated by a SW robot using a RPA tool is transferring data from one system to another via their respective UIs, e.g., copying records from a spreadsheet application into a web-based enterprise information system [21]. In recent years, much progress has been made both in terms of research and technical development on RPA, resulting in many deployments for industrial-oriented services [19, 4, 17, 28]. Moreover, the market of RPA solutions has developed rapidly and today includes more than 50 vendors developing tools that provide SW robots with advanced functionalities for automating office tasks in operations like accounting and customer service [5]. Nonetheless, when considering state-of-the-art RPA technology, it becomes apparent that the current generation of RPA tools is driven by predefined rules and manual configurations made by expert users rather than automated techniques [2, 10].

In fact, as reported in [16], in the early stages of the RPA life-cycle it is required the support of skilled human experts to: *(i)* identify the candidate routines to automate by means of interviews and observation of workers conducting their daily work, *(ii)* record the interactions that take place during routines' enactment on the UI of software applications into dedicated *UI logs*, and *(iii)* manually specify their conceptual and technical structure (often in form of flowchart diagrams) for defining the behavior of SW robots.

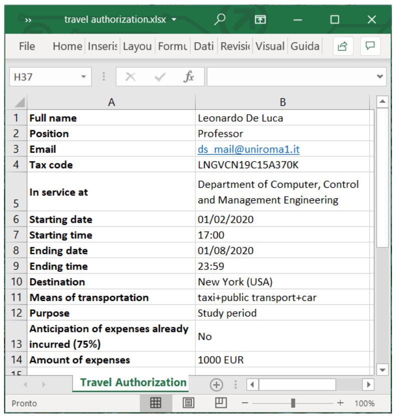
This approach is not effective in case of UI logs that keep track of many routines executions, since the designer should have a global vision of all possible variants of the routines to define the appropriate behaviours of SW robots, which becomes complicated when the number of variants increases. Indeed, in presence of UI logs that collect information about several routines, the recorded actions are mixed in some order that reflects the particular order of their execution by the user, making the identification of candidate routines in a UI log a time-consuming and error-prone task. The challenge to understand which user actions contribute to which routines inside a UI log is known as *segmentation* [2, 21].

This paper aims to explore the challenge of automated segmentation in RPA through the evaluation of two research questions: *(i)* what are the variants of a segmentation solution needed to properly deal with different kinds of UI log? *(ii)* to what extent such variants are supported by literature approaches?

To answer these research questions, we first leverage a concrete RPA use case in the administrative sector to explain the segmentation issue (Section 2). After having described the relevant background on UI logs (Section 3), we discuss how a segmentation technique should behave in presence of three different (and relevant) forms of UI logs, which may consist of: *(i)* several executions of the same routine, *(ii)* several executions of many routines without the possibility to have user actions in common, and *(iii)* several executions of many routines with the possibility to have user actions in common (Section 4). Then, we investigate how and if such forms of UI logs are tackled by the current state-of-the-art segmentation approaches (Section 5). Finally, we provide future directions for automated segmentation in RPA based on the findings of our study (Section 6).

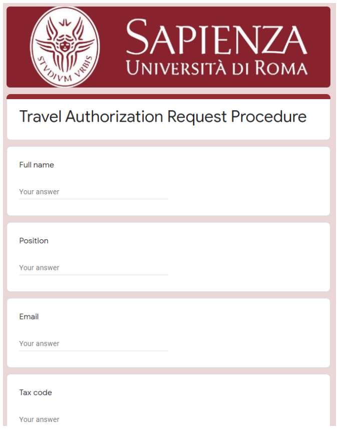
2 A RPA Use Case

In this section, we describe a RPA use case derived by a real-life scenario at Department of Computer, Control and Management Engineering (DIAG) of Sapienza Universitá di Roma. The scenario concerns the filling of the travel authorization request form made by personnel 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 (e.g., to request in advance a visa). When ready, the spreadsheet is



	A	B
1	**Full name**	Leonardo De Luca
2	**Position**	Professor
3	**Email**	ds_mail@uniroma1.it
4	**Tax code**	LNGVCN19C15A370K
5	**In service at**	Department of Computer, Control and Management Engineering
6	**Starting date**	01/02/2020
7	**Starting time**	17:00
8	**Ending date**	01/08/2020
9	**Ending time**	23:59
10	**Destination**	New York (USA)
11	**Means of transportation**	taxi+public transport+car
12	**Purpose**	Study period
13	**Anticipation of expenses already incurred (75%)**	No
14	**Amount of expenses**	1000 EUR

(a) Excel spreadsheet



(b) Google form

Fig. 1: UIs involved in the use case

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. Fig. 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.

In addition, if the request applicant declares that s/he would like to use her/his personal car as one of the means of transport for the travel, then s/he has to fill a dedicated web form required for activating a special insurance for the part of the travel that will be performed with the car. This further request will be delivered to the Administration staff via email, and the employee in charge of processing it can either approve or reject such request. At the end, the applicant will be automatically notified via email of the approval/rejection of the request.

The above procedure, which involves two main routines (in the following, we will denote them as R1 and R2), is performed manually by an employee of the Administration Office of DIAG, and it should be repeated for any new travel request. Routines such as these ones are good candidates to be encoded with executable scripts and enacted by means of a SW robot within a commercial RPA tool. However, unless there is complete a-priori knowledge of the specific routines that are enacted on the UI and of their concrete composition, their automated identification from an UI log is challenging, since the associated user actions may be scattered across the log, interleaved with other actions that are not part of the routine under analysis, and potentially shared by many routines.

For the sake of understandability, we show in figures 2 and 3 the interaction models of R1 and R2 required to represent the structure of the routines of

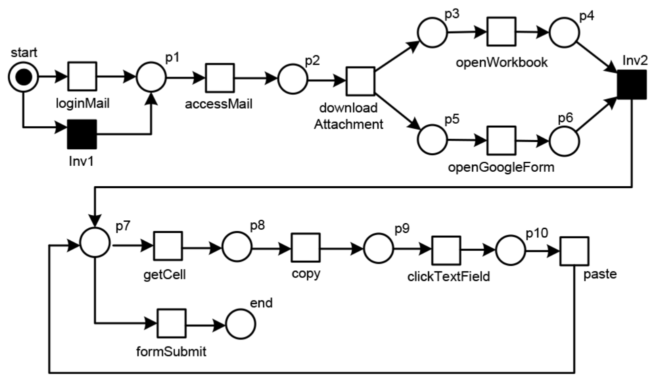


Fig. 2: Interaction model for R1

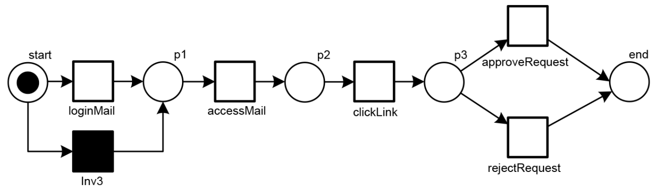


Fig. 3: Interaction model for R2

interest, depicted as Petri nets.[1] For example, analyzing the Petri net in Fig. 2, it becomes clear that a proper execution of R1 requires a path on the UI made by the following user actions:[2]

- 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;
- 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 a text field of the Google form;
- formSubmit, to finally submit the Google form to the internal database.

As shown in Fig. 2, the user actions openWorkbook and openGoogleForm can be performed in any order. Moreover, the sequence of actions ⟨getCell, copy,

[1] The research literature is rich of notations for expressing human-computer dialogs as interaction models. Among them, Petri nets guarantee a good trade-off between expressiveness and understandability of the models [26].

[2] Note that the user actions recorded in a UI log can have a finer granularity than the high-level ones used here just with the purpose of describing the routine's behaviour.

clickTextField, paste⟩ will be repeated for any travel information to be moved from the Excel spreadsheet to the Google form. On the other hand, the path of user actions in the UI to properly enact R2 is as follows:

– loginMail, to access the client email;
– accessMail, to access the specific email with the request for travel insurance;
– clickLink, to click the link included in the email that opens the Google form with the request to activate the travel insurance on a web browser;
– approveRequest, to press the button on the Google form that approves the request;
– rejectRequest, to press the button on the Google form that rejects the request;

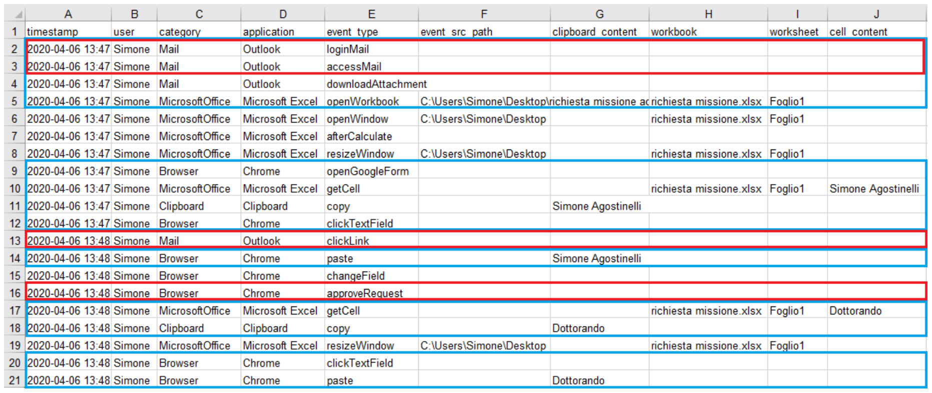
Note that the execution of approveRequest and rejectRequest is exclusive, cf. the Petri net in Fig. 3. Then, in the interaction models of R1 and R2, there are transitions that do not represent user actions but are needed to correctly represent the structure of such models. These transitions, drawn with a black-filled rectangle, are said to be "invisible", and are not recorded in the UI logs (cf. Inv1, Inv2 and Inv3).

3 Preliminaries on User Interface Logs

In this section, we provide some preliminary notions about User Interface (UI) logs, needed to understand the rest of the paper. A single UI log in its raw form consists of a long sequence of user actions recorded during one user session.[3] Such actions include all the steps required to accomplish one or more relevant routines using the UI of one or many sw application/s. For instance, in Fig. 4, we show a snapshot of a UI log captured using a dedicated action logger[4] during the execution of R1 and R2. The employed action logger enables to record the *events* happened on the UI, enriched with several data fields describing their "anatomy". For a given event, such fields are useful to keep track the name and the timestamp of the user action performed on the UI, the involved sw application, the human/sw resource that performed the action, etc.

For the sake of understandability, we assume here that any user action associated to each event recorded in the UI log is mapped at most with one (and only one) Petri net transition, and that the collection of labels associated to the Petri net transitions is defined over the same alphabet as the user actions in the UI log,[5] i.e., the alphabet of user actions in the UI log is a *superset* of that used for defining the labels of Petri net transitions. In the running example, we can recognize in R1 and R2 a universe of user actions of interest $Z = \{$loginMail, accessMail, downloadAttachment, openWorkbook, openGoogleForm, getCell, copy, clickTextField, paste, formSubmit, clickLink, approveRequest, rejectRequest$\}$.

[3] We interpret a user session as a group of interactions that a single user takes within a given time frame on the UI of a specific computer system.
[4] The working of the action logger is described in [1]. The tool is available at: https://github.com/bpm-diag/smartRPA
[5] In [22], it is shown how these assumptions can be removed.



	A	B	C	D	E	F	G	H	I	J
1	timestamp	user	category	application	event_type	event_src_path	clipboard_content	workbook	worksheet	cell_content
2	2020-04-06 13:47	Simone	Mail	Outlook	loginMail					
3	2020-04-06 13:47	Simone	Mail	Outlook	accessMail					
4	2020-04-06 13:47	Simone	Mail	Outlook	downloadAttachment					
5	2020-04-06 13:47	Simone	MicrosoftOffice	Microsoft Excel	openWorkbook	C:\Users\Simone\Desktop\richiesta missione ac		richiesta missione.xlsx	Foglio1	
6	2020-04-06 13:47	Simone	MicrosoftOffice	Microsoft Excel	openWindow	C:\Users\Simone\Desktop		richiesta missione.xlsx	Foglio1	
7	2020-04-06 13:47	Simone	MicrosoftOffice	Microsoft Excel	afterCalculate					
8	2020-04-06 13:47	Simone	MicrosoftOffice	Microsoft Excel	resizeWindow	C:\Users\Simone\Desktop		richiesta missione.xlsx	Foglio1	
9	2020-04-06 13:47	Simone	Browser	Chrome	openGoogleForm					
10	2020-04-06 13:47	Simone	MicrosoftOffice	Microsoft Excel	getCell			richiesta missione.xlsx	Foglio1	Simone Agostinelli
11	2020-04-06 13:47	Simone	Clipboard	Clipboard	copy		Simone Agostinelli			
12	2020-04-06 13:47	Simone	Browser	Chrome	clickTextField					
13	2020-04-06 13:48	Simone	Mail	Outlook	clickLink					
14	2020-04-06 13:48	Simone	Browser	Chrome	paste		Simone Agostinelli			
15	2020-04-06 13:48	Simone	Browser	Chrome	changeField					
16	2020-04-06 13:48	Simone	Browser	Chrome	approveRequest					
17	2020-04-06 13:48	Simone	MicrosoftOffice	Microsoft Excel	getCell			richiesta missione.xlsx	Foglio1	Dottorando
18	2020-04-06 13:48	Simone	Clipboard	Clipboard	copy		Dottorando			
19	2020-04-06 13:48	Simone	MicrosoftOffice	Microsoft Excel	resizeWindow	C:\Users\Simone\Desktop		richiesta missione.xlsx	Foglio1	
20	2020-04-06 13:48	Simone	Browser	Chrome	clickTextField					
21	2020-04-06 13:48	Simone	Browser	Chrome	paste		Dottorando			

Fig. 4: Snapshot of a UI log captured during the executions of R1 and R2

As shown in Fig. 4, a UI log is not specifically recorded to capture pre-identified routines. A UI log may contain multiple and interleaved executions of one/many routine/s (cf. the blue/red boxes that group the user actions belonging to R1 and R2, respectively), as well as redundant behavior and noise. We consider as *redundant* any action that is unnecessary repeated during the execution of a routine, e.g., a text value that is first pasted in a wrong field and then is moved in the right place through a corrective action on the UI. On the other hand, we consider as *noise* all those actions that do not contribute to the achievement of any routine target, e.g., a window that is resized. In Fig. 4, the sequences of user actions that are not surrounded by a blue/red box can be safely labeled as noise.

Segmentation techniques aim to extract from a UI log all those user actions of a routine R, filtering out redundant actions and noise. Any sequence of actions in the UI log that can be replayed from the initial to the final marking of the Petri net-based interaction model of R is said to be a *routine trace* of R. For example, a valid routine trace of R1 is ⟨loginMail, accessMail, downloadAttachment, openWorkbook, openGoogleForm, getCell, copy, clickTextField, paste, formSubmit⟩. The interaction model of R1 suggests that valid routine traces 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. On the other hand, two main routine traces can be extracted from R2: ⟨loginMail, accessMail, clickLink, acceptRequest⟩ and ⟨loginMail, accessMail, clickLink, rejectRequest⟩, again with the possibility to skip loginMail, i.e., the access to the client email.

4 Identifying the Segmentation Cases

Given a UI log that consists of events including user actions with the same granularity[6] and potentially belonging to different routines, in the RPA domain

[6] 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.).

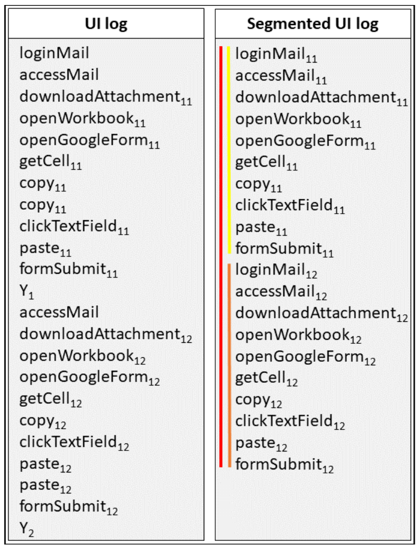
segmentation is the task of clustering parts of the log together which belong to the same routine. In a nutshell, the challenge is to automatically understand which user actions contribute to which routines, and organize such user actions in well bounded routine traces [2, 21].

As shown in Section 3, in general a UI log stores information about several routines enacted in an interleaved fashion, with the possibility that a specific user action is shared by different routines. Furthermore, actions providing redundant behavior or not belonging to any of the routine under observation may be recorded in the log, generating noise that should be filtered out by a segmentation technique. Based on the above considerations, and on a concrete analysis of real UI logs recorded during the enactment of the routines presented in Section 2, we have identified three main forms of UI logs, which can be categorized according to the fact that: *(i)* any user action in the log exclusively belongs to a specific routine (Case 1); *(ii)* the log records the execution of many routines that do not have any user action in common (Case 2); *(iii)* the log records the execution of many routines, and the possibility exists that some performed user actions are shared by many routines at the same time (Case 3). In the following, we analyze the characteristics of the three cases and of their variants. For the sake of understandability, we use a numerical subscript ij associated to any user action to indicate that it belongs to the $j-th$ execution of the $i-th$ routine under study. Of course, this information is not recorded in the UI log, and discovering it (i.e., the identification of the subscripts) is one of the "implicit" effects of segmentation when routine traces are built.

Case 1. This is the case when a UI log captures many executions of the same routine. Of course, in this scenario it is not possible to distinguish between shared and non-shared user actions by different routines, since the UI log keeps track only of executions associated to a single routine. Two main variants exist:

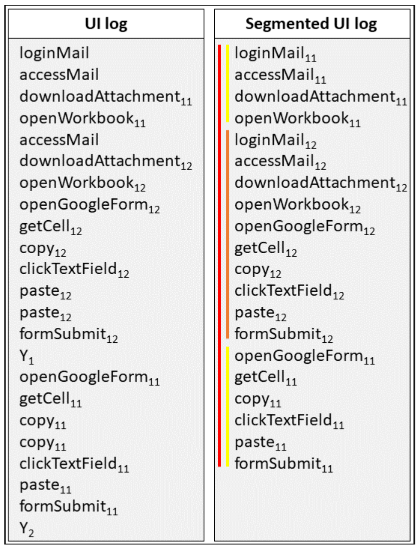
- **Case 1.1**. Starting from the use case in Section 2, let us consider the case of a UI log that records a sequence of user actions resulting from many non-interleaved executions of R1 (cf. Fig. 5(a)). We have also the presence of some user actions that potentially belong *at the same time* to many executions of the routine itself. This is the case of loginMail, which can be performed exactly once at the beginning of a user session and can be "shared" by many executions of the same routine. Applying a segmentation technique to the above UI log would trivially produce a segmented UI log where the (already well bounded) executions of R1 are organized as different routine traces: the yellow and orange vertical lines outline the routine traces, while the red line outlines the routine segment of R1.
- **Case 1.2**. The same segmented UI log is obtained when the executions of R1 are recorded in an interleaved fashion in the original UI log (cf. Fig. 5(b)). Here, the segmentation task is more challenging, because the user actions of different executions of the same routine are interleaved among each others, and it is not known a-priori to which execution they belong.

We will discuss the abstraction issue in Section 5, where state-of-the-art techniques are shown that enable to flatten the content of a log to a same granularity level.



UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
copy$_{11}$	clickTextField$_{11}$
clickTextField$_{11}$	paste$_{11}$
paste$_{11}$	formSubmit$_{11}$
formSubmit$_{11}$	loginMail$_{12}$
Y$_{1}$	accessMail$_{12}$
accessMail	downloadAttachment$_{12}$
downloadAttachment$_{12}$	openWorkbook$_{12}$
openWorkbook$_{12}$	openGoogleForm$_{12}$
openGoogleForm$_{12}$	getCell$_{12}$
getCell$_{12}$	copy$_{12}$
copy$_{12}$	clickTextField$_{12}$
clickTextField$_{12}$	paste$_{12}$
paste$_{12}$	formSubmit$_{12}$
paste$_{12}$	
formSubmit$_{12}$	
Y$_{2}$	

(a) Case 1.1



UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
accessMail	loginMail$_{12}$
downloadAttachment$_{12}$	accessMail$_{12}$
openWorkbook$_{12}$	downloadAttachment$_{12}$
openGoogleForm$_{12}$	openWorkbook$_{12}$
getCell$_{12}$	openGoogleForm$_{12}$
copy$_{12}$	getCell$_{12}$
clickTextField$_{12}$	copy$_{12}$
paste$_{12}$	clickTextField$_{12}$
paste$_{12}$	paste$_{12}$
formSubmit$_{12}$	formSubmit$_{12}$
Y$_{1}$	openGoogleForm$_{11}$
openGoogleForm$_{11}$	getCell$_{11}$
getCell$_{11}$	copy$_{11}$
copy$_{11}$	clickTextField$_{11}$
copy$_{11}$	paste$_{11}$
clickTextField$_{11}$	formSubmit$_{11}$
paste$_{11}$	
formSubmit$_{11}$	
Y$_{2}$	

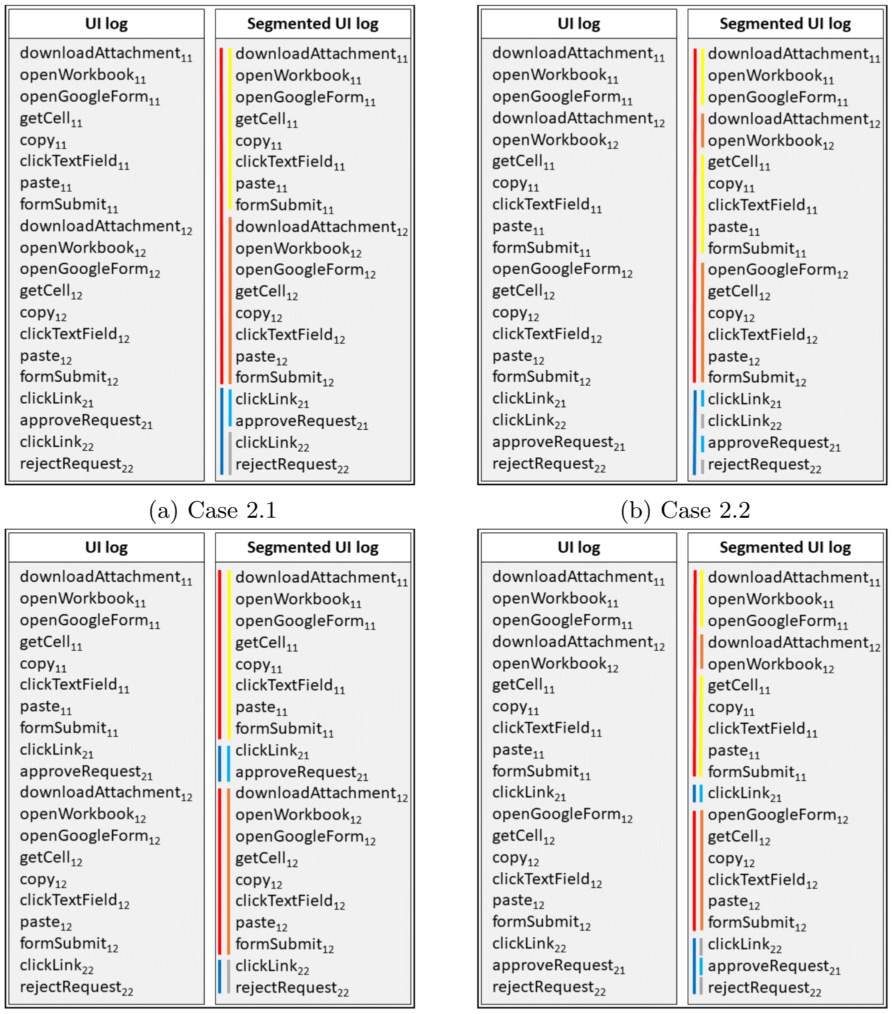
(b) Case 1.2

Fig. 5: Variants for Case 1

Both variants of Case 1 are affected by *noise* or *redundant* actions. The logs contain elements of noise, i.e., user actions $\boldsymbol{Y_{k\in\{1,n\}}} \in Z$ (remind that Z is the universe of user actions allowed by a UI log, as introduced in Section 3) that are not allowed by R1, and redundant actions like copy and paste that are unnecessary repeated multiple times. Noise and redundant actions need to be filtered out during the segmentation task because they do not contribute to the achievement of the routine's target. For the sake of space, in the following analysis we do not consider anymore the presence of noise and redundant actions, since their handling is similar for all the cases.

Case 2. In this case, a UI log captures many executions of different routines, with the assumption that the interaction models of such routines include only transitions associated to user actions that are exclusive for that routines. To comply with the latter constraint, let us suppose that in both interaction models of R1 and R2 the transitions loginMail and accessMail are not required. Four main variants of Case 2 can be identified:

– **Case 2.1**. Let us consider the UI log in Fig. 6(a). The output of the segmentation task would consist of a segmented log where the (already well bounded) executions of R1 and R2 are organized as different routine traces: *(i)* the yellow and orange vertical lines outline the routine traces of R1, *(ii)* the light blue and grey vertical lines outline the routine traces of R2, while *(iii)* the outer red and blue lines outline the routine segments of R1 and R2. In the following, the coloring scheme will be kept the same.



UI log	Segmented UI log
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
downloadAttachment$_{12}$	downloadAttachment$_{12}$
openWorkbook$_{12}$	openWorkbook$_{12}$
openGoogleForm$_{12}$	openGoogleForm$_{12}$
getCell$_{12}$	getCell$_{12}$
copy$_{12}$	copy$_{12}$
clickTextField$_{12}$	clickTextField$_{12}$
paste$_{12}$	paste$_{12}$
formSubmit$_{12}$	formSubmit$_{12}$
clickLink$_{21}$	clickLink$_{21}$
approveRequest$_{21}$	approveRequest$_{21}$
clickLink$_{22}$	clickLink$_{22}$
rejectRequest$_{22}$	rejectRequest$_{22}$

(a) Case 2.1

UI log	Segmented UI log
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
downloadAttachment$_{12}$	downloadAttachment$_{12}$
openWorkbook$_{12}$	openWorkbook$_{12}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
openGoogleForm$_{12}$	openGoogleForm$_{12}$
getCell$_{12}$	getCell$_{12}$
copy$_{12}$	copy$_{12}$
clickTextField$_{12}$	clickTextField$_{12}$
paste$_{12}$	paste$_{12}$
formSubmit$_{12}$	formSubmit$_{12}$
clickLink$_{21}$	clickLink$_{21}$
clickLink$_{22}$	clickLink$_{22}$
approveRequest$_{21}$	approveRequest$_{21}$
rejectRequest$_{22}$	rejectRequest$_{22}$

(b) Case 2.2

UI log	Segmented UI log
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
clickLink$_{21}$	clickLink$_{21}$
approveRequest$_{21}$	approveRequest$_{21}$
downloadAttachment$_{12}$	downloadAttachment$_{12}$
openWorkbook$_{12}$	openWorkbook$_{12}$
openGoogleForm$_{12}$	openGoogleForm$_{12}$
getCell$_{12}$	getCell$_{12}$
copy$_{12}$	copy$_{12}$
clickTextField$_{12}$	clickTextField$_{12}$
paste$_{12}$	paste$_{12}$
formSubmit$_{12}$	formSubmit$_{12}$
clickLink$_{22}$	clickLink$_{22}$
rejectRequest$_{22}$	rejectRequest$_{22}$

(c) Case 2.3

UI log	Segmented UI log
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
downloadAttachment$_{12}$	downloadAttachment$_{12}$
openWorkbook$_{12}$	openWorkbook$_{12}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
clickLink$_{21}$	clickLink$_{21}$
openGoogleForm$_{12}$	openGoogleForm$_{12}$
getCell$_{12}$	getCell$_{12}$
copy$_{12}$	copy$_{12}$
clickTextField$_{12}$	clickTextField$_{12}$
paste$_{12}$	paste$_{12}$
formSubmit$_{12}$	formSubmit$_{12}$
clickLink$_{22}$	clickLink$_{22}$
approveRequest$_{21}$	approveRequest$_{21}$
rejectRequest$_{22}$	rejectRequest$_{22}$

(d) Case 2.4

Fig. 6: Variants for Case 2

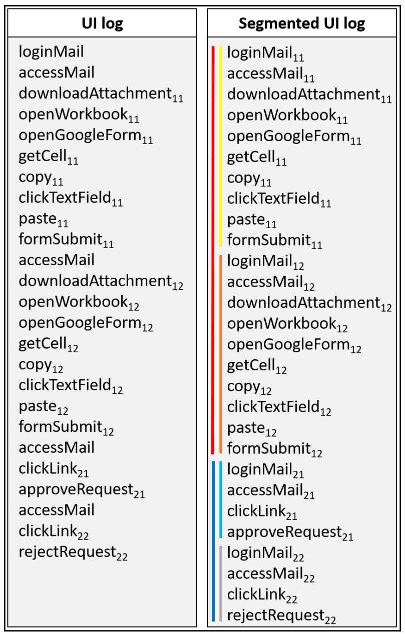
- **Case 2.2**. Similarly to what already seen in Case 1.2, it may happen that many executions of the same routine are interleaved among each other (cf. Fig. 6(b)), e.g., the first execution of R1 is interleaved with the second execution of R1, the first execution of R2 is interleaved with the second execution of R2, and so on.
- **Case 2.3**. Another variant is when the UI log records in an interleaved fashion many different routines but not the routine executions (cf. Fig. 6(c)), e.g., the first execution of R2 follows the first execution of R1, the second execution of R2 follows the second execution of R1, and so on.

– **Case 2.4**. The complexity of the segmentation task becomes more challenging in presence of both interleaved routines and routine executions (cf. Fig. 6(d)), e.g., the first execution of R1 is interleaved with the second execution of R1; the second execution of R1 is interleaved with the first execution of R2; the first execution of R2 is interleaved with the second execution of R2.

Case 3. In this case, a UI log captures many executions of different routines, and there exist user actions that are shared by such routines. This case perfectly reflects what happens in the use case of Section 2. In particular, loginMail and accessMail are shared by R1 and R2, as they are included in the interaction models of both routines. Four variants can be distinguished:

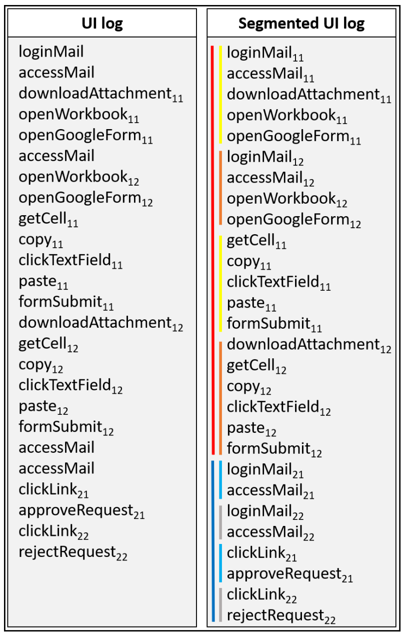
– **Case 3.1**. Let us consider the UI log depicted in Fig. 7(a). loginMail is *potentially involved* in the enactment of any execution of R1 and R2, while accessMail is *required by all* executions of R1 and R2, but it is not clear the association between the single executions of accessMail and the routine executions they belong to. The complexity of the segmentation task here lies in understanding to which routine traces the execution of loginMail and accessMail belong to. The outcome of the segmentation task will be a segmented log where the executions of R1 and R2 are organized as different routine traces according to the coloring scheme explained in Case 2.1.
– **Case 3.2**. This is the case when the UI log records interleaved executions of the same routine in presence of shared user actions (cf. Fig. 7(b)), e.g., the first execution of R1 is interleaved with the second execution of R1, and the first execution of R2 is interleaved with the second execution of R2.
– **Case 3.3**. Another variant is when the UI log records in an interleaved fashion many different routines but not the routine executions in presence of shared user actions (cf. Fig. 7(c)), e.g.: the first execution of R2 follows the first execution of R1 and the second execution of R2 follows the second execution of R1.
– **Case 3.4**. The segmentation task becomes more challenging in presence of more complex UI logs consisting of both interleaved routines and routine executions with shared user actions (cf. Fig. 7(d)), e.g., the first execution of R1 is interleaved with the second execution of R1, the second execution of R1 is interleaved with the first execution of R2, and the first execution of R2 is interleaved with the second execution of R2.

The above three cases and their variants have in common that all the user actions are stored within a single UI log. It may happen that the same routine is spread across multiple UI logs, in particular when there are multiple users that are involved in the execution of the routine on different computer systems. This case can be tackled by "merging" the UI logs where the routine execution is distributed into a single UI log, reducing the segmentation issue to one of the already analyzed cases. It is worth noticing that although the classification of cases and variants was illustrated with only two routines (interleaving or not), the classification is defined in a generic way and applies to any number of routines.



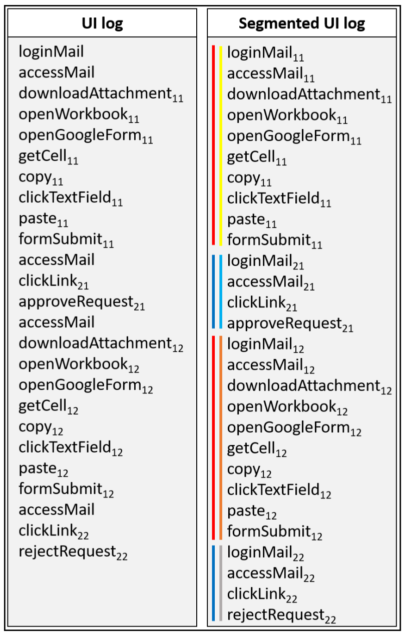
UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
accessMail	loginMail$_{12}$
downloadAttachment$_{12}$	accessMail$_{12}$
openWorkbook$_{12}$	downloadAttachment$_{12}$
openGoogleForm$_{12}$	openWorkbook$_{12}$
getCell$_{12}$	openGoogleForm$_{12}$
copy$_{12}$	getCell$_{12}$
clickTextField$_{12}$	copy$_{12}$
paste$_{12}$	clickTextField$_{12}$
formSubmit$_{12}$	paste$_{12}$
accessMail	formSubmit$_{12}$
clickLink$_{21}$	loginMail$_{21}$
approveRequest$_{21}$	accessMail$_{21}$
accessMail	clickLink$_{21}$
clickLink$_{22}$	approveRequest$_{21}$
rejectRequest$_{22}$	loginMail$_{22}$
	accessMail$_{22}$
	clickLink$_{22}$
	rejectRequest$_{22}$

(a) Case 3.1



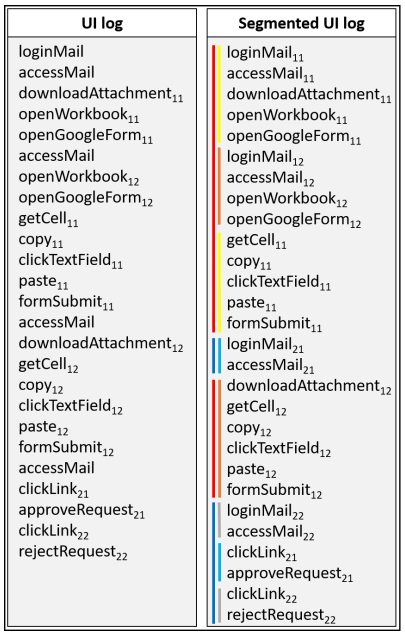
UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
accessMail	loginMail$_{12}$
openWorkbook$_{12}$	accessMail$_{12}$
openGoogleForm$_{12}$	openWorkbook$_{12}$
getCell$_{11}$	openGoogleForm$_{12}$
copy$_{11}$	getCell$_{11}$
clickTextField$_{11}$	copy$_{11}$
paste$_{11}$	clickTextField$_{11}$
formSubmit$_{11}$	paste$_{11}$
downloadAttachment$_{12}$	formSubmit$_{11}$
getCell$_{12}$	downloadAttachment$_{12}$
copy$_{12}$	getCell$_{12}$
clickTextField$_{12}$	copy$_{12}$
paste$_{12}$	clickTextField$_{12}$
formSubmit$_{12}$	paste$_{12}$
accessMail	formSubmit$_{12}$
accessMail	loginMail$_{21}$
clickLink$_{21}$	accessMail$_{21}$
approveRequest$_{21}$	loginMail$_{22}$
clickLink$_{22}$	accessMail$_{22}$
rejectRequest$_{22}$	clickLink$_{21}$
	approveRequest$_{21}$
	clickLink$_{22}$
	rejectRequest$_{22}$

(b) Case 3.2



UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
getCell$_{11}$	getCell$_{11}$
copy$_{11}$	copy$_{11}$
clickTextField$_{11}$	clickTextField$_{11}$
paste$_{11}$	paste$_{11}$
formSubmit$_{11}$	formSubmit$_{11}$
accessMail	loginMail$_{21}$
clickLink$_{21}$	accessMail$_{21}$
approveRequest$_{21}$	clickLink$_{21}$
accessMail	approveRequest$_{21}$
downloadAttachment$_{12}$	loginMail$_{12}$
openWorkbook$_{12}$	accessMail$_{12}$
openGoogleForm$_{12}$	downloadAttachment$_{12}$
getCell$_{12}$	openWorkbook$_{12}$
copy$_{12}$	openGoogleForm$_{12}$
clickTextField$_{12}$	getCell$_{12}$
paste$_{12}$	copy$_{12}$
formSubmit$_{12}$	clickTextField$_{12}$
accessMail	paste$_{12}$
clickLink$_{22}$	formSubmit$_{12}$
rejectRequest$_{22}$	loginMail$_{22}$
	accessMail$_{22}$
	clickLink$_{22}$
	rejectRequest$_{22}$

(c) Case 3.3



UI log	Segmented UI log
loginMail	loginMail$_{11}$
accessMail	accessMail$_{11}$
downloadAttachment$_{11}$	downloadAttachment$_{11}$
openWorkbook$_{11}$	openWorkbook$_{11}$
openGoogleForm$_{11}$	openGoogleForm$_{11}$
accessMail	loginMail$_{12}$
openWorkbook$_{12}$	accessMail$_{12}$
openGoogleForm$_{12}$	openWorkbook$_{12}$
getCell$_{11}$	openGoogleForm$_{12}$
copy$_{11}$	getCell$_{11}$
clickTextField$_{11}$	copy$_{11}$
paste$_{11}$	clickTextField$_{11}$
formSubmit$_{11}$	paste$_{11}$
accessMail	formSubmit$_{11}$
downloadAttachment$_{12}$	loginMail$_{21}$
getCell$_{12}$	accessMail$_{21}$
copy$_{12}$	downloadAttachment$_{12}$
clickTextField$_{12}$	getCell$_{12}$
paste$_{12}$	copy$_{12}$
formSubmit$_{12}$	clickTextField$_{12}$
accessMail	paste$_{12}$
clickLink$_{21}$	formSubmit$_{12}$
approveRequest$_{21}$	loginMail$_{22}$
clickLink$_{22}$	accessMail$_{22}$
rejectRequest$_{22}$	clickLink$_{21}$
	approveRequest$_{21}$
	clickLink$_{22}$
	rejectRequest$_{22}$

(d) Case 3.4

Fig. 7: Variants for Case 3

Table 1: Literature approaches to tackle segmentation variants

Papers	**Case 1**		**Case 2**				**Case 3**			
	1.1	1.2	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4
Agostinelli *et al.* [3]	✓	✓	✓	✓	✓	✓	∼	∼		
Baier *et al.* [7]	✓		✓							
Bayomie *et al.* [8]	✓									
Bosco *et al.* [9]	✓		✓							
Kumar *et al.* [18]	✓		✓							
Leno *et al.* [20]	✓	∼	✓	∼	∼	∼				
Liu [23]	✓		✓	∼	∼	∼				
Fazzinga *et al.* [12]	✓		✓				✓			
Ferreira *et al.* [13]	✓		✓							
Mannhardt *et al.* [24]	✓		✓							
Măruşter *et al.* [27]	✓									
Srivastava *et al.* [29]	✓		✓							

5 Assessing the Segmentation Approaches

In the field of RPA, segmentation is an issue still not so explored, since the current practice adopted by commercial RPA tools for identifying the routine steps often consists of detailed observations of workers conducting their daily work. Such observations are then "converted" in explicit flowchart diagrams [16], which are manually modeled by expert RPA analysts to depict all the potential behaviours (i.e., the traces) of a specific routine. In this setting, as the routine traces have been already (implicitly) identified, segmentation can be neglected.

On the other hand, following a similar trend that has been occurring in the BPM domain [25], the research on RPA is moving towards the application of intelligent techniques to automate all the steps of a RPA project, as proven by many recent works in this direction (see below). In this context, segmentation can be considered as one of the "hot" key research effort to investigate [2, 21].

Table 1 summarizes the current literature techniques that could be leveraged to tackle the different variants of the segmentation issue. We will use ✓ to denote the full ability of an approach to deal with a specific UI log variant, while ∼ denotes that the approach is only partially able to deal with a specific UI log variant (i.e., under certain conditions). In the following, we discuss to what extent such variants can be supported by existing literature approaches, grouping them by means of their research area. It is worth noticing that the assessment of the literature approaches is based on what was reported in the associated papers.

Concerning RPA-related techniques, Bosco et al. [9] 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. This approach is effective in case of UI logs that keep track of well-bounded routine executions (Case 1.1 and Case 2.1), and becomes inadequate when the UI log records information about several routines whose actions are potentially interleaved. In this direction, Leno et al. [20] 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 [20] is able to achieve cases 1.1, 1.2 and 2.1, and partially cases 2.2, 2.3 and 2.4, but (for the latter) it loses in accuracy in presence of recurrent noise and interleaved routine executions. The main limitation of the above techniques is tackled in [3]. Here, a supervised segmentation algorithm able to achieve all variants of cases 1, 2 and (partially) 3 has been proposed, except when there are interleaved executions of shared user actions of many routines. In that case, the risk exists that a shared user action is associated to a wrong routine execution (i.e., Case 3.3 and Case 3.4 are not covered). However, to make the algorithm works, it is required to know a-priori the structure of the interaction models of the routines to identify in the UI log.

Even if more focused on traditional business processes in BPM rather than on RPA routines, Bayomie et al. [8] 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, as a consequence this technique is able to achieve Case 1.1 only. In the field of process discovery, Măruşter et al. [27] propose an empirical method for inducing rule sets from event logs containing execution of one process only. Therefore, as in [8], this method is able to achieve Case 1.1 only, thus making the technique ineffective in 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 is not able to discriminate which event instance (but just the event type) belongs to which process model. This makes [12] effective to tackle Case 1.1, Case 2.1 and Case 3.1. Closely related to [12], there is the work of Liu [23]. 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 achieve both cases 1.1 and 2.1, but partially cases 2.2, 2.3 and 2.4 (the approach can lose accuracy in assigning operations to workflows).

Differently from the previous works, Time-Aware Partitioning (TAP) techniques cut event logs on the basis of the temporal distance between two events [29, 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 are not able to handle neither interleaved user actions of different routine executions nor interleaved user actions of different routines. As a consequence, TAP techniques are able to achieve cases 1.1 and 2.1.

There exist other approaches whose the target is not to exactly resolve the segmentation issue. Many research works exist that analyze UI logs at different levels of abstraction and that can be potentially useful to realize 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. [7] 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. [24] present a technique that 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). 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). As a consequence, all abstraction techniques are effective to achieve Case 1.1 and Case 2.1 only.

6 Conclusion

In this work, we have leveraged a real-life use case in the administrative sector to explore the issue of automated segmentation in RPA, detecting its potential variants and discussing to what extent the literature approaches are able to support such variants. 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 few works [12, 3, 20, 23] 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 yet infrequent variants of a routine may exist in the UI log. For this reason, we think that an important first step towards the development of a more complete segmentation technique is to shift from the current model-based approaches to learning-based ones. In this direction, two main strategies seem feasible to relax the above supervised assumption: *(i)* to investigate *sequential pattern mining* techniques [11] to examine frequent sequences of user actions having common data attributes; *(ii)* to employ clustering techniques to aggregate user actions into clusters, where any cluster represents a particular routine and each associated sequence of user actions a routine trace [15, 14].

Finally, we want to underline that process discovery techniques [6] can also play a relevant role to tackle the segmentation issue, as demonstrated by some literature works [23, 12, 8] discussed in Section 5. However, the issue is that the majority of 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 and without any clear starting/ending point. Thus, a UI log is more similar to a unique (long) 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.

Acknowledgments. This work has been supported by the "Dipartimento di Eccellenza" grant, the H2020 project DataCloud and the Sapienza grant BPbots.

References

1. Agostinelli, S., Lupia, M., Marrella, A., Mecella, M.: Automated Generation of Executable RPA Scripts from User Interface Logs. In: 18th Int. Conf. on Business Process Management (RPA Forum). pp. 116–131 (2020)
2. Agostinelli, S., Marrella, A., Mecella, M.: Research Challenges for Intelligent Robotic Process Automation. In: 17th Int. Conf. on Business Process Management (Workshop on Artificial Intelligence). pp. 12–18 (2019)
3. Agostinelli, S., Marrella, A., Mecella, M.: Automated Segmentation of User Interface Logs. In: Robotic Process Automation. Management, Technology, Applications. De Gruyter (2021)
4. Aguirre, S., Rodriguez, A.: Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. In: Applied Computer Sciences in Engineering - 4th Workshop on Engineering Applications. pp. 65–71 (2017)
5. AI-Multiple: All 55 RPA Software Tools & Vendors of 2021: Sortable List. https://blog.aimultiple.com/rpa-tools/ (2021)
6. 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)
7. Baier, T., Rogge-Solti, A., Mendling, J., Weske, M.: Matching of Events and Activities: an Approach Based on Behavioral Constraint Satisfaction. In: 30th ACM Symp. on Applied Computing. pp. 1225–1230 (2015)
8. 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)
9. Bosco, A., Augusto, A., Dumas, M., La Rosa, M., Fortino, G.: Discovering Automatable Routines from User Interaction Logs. In: 17th Int. Conf. on Business Process Management (Forum track). pp. 144–162 (2019)

10. Chakraborti, T., Isahagian, V., Khalaf, R., Khazaeni, Y., Muthusamy, V., Rizk, Y., Unuvar, M.: From Robotic Process Automation to Intelligent Process Automation: Emerging Trends. In: 18th Int. Conf. on Business Process Management (RPA Forum). pp. 215–228 (2020)
11. Dong, G., Pei, J.: Sequence Data Mining, Advances in Database Systems, vol. 33. Kluwer (2007)
12. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: Efficiently Interpreting Traces of Low Level Events in Business Process Logs. Inf. Syst. **73**, 1–24 (2018)
13. Ferreira, D.R., Szimanski, F., Ralha, C.G.: Improving Process Models by Mining Mappings of Low-Level Events to High-Level Activities. Inf. Syst. **43**(2), 379–407 (2014)
14. Folino, F., Guarascio, M., Pontieri, L.: Mining Predictive Process Models out of Low-Level Multidimensional Logs. In: 26th Int. Conf. on Advanced Information System Engineering. pp. 533–547 (2014)
15. Günther, C.W., Rozinat, A., van Der Aalst, W.M.: Activity Mining by Global Trace Segmentation. In: 7th Int. Conf. on Business Process Management (Workshops). pp. 128–139 (2009)
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: 31th Int. Conf. on Advanced Information System Engineering. pp. 446–461 (2019)
17. Kokina, J., Blanchette, S.: Early Evidence of Digital Labor in Accounting: Innovation with Robotic Process Automation. Int. J. Account. Inf. Syst. **35** (2019)
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)
19. Lacity, M., Willcocks, L.: Robotic Process Automation at Telefónica O2. MIS Q. Executive **15** (2016)
20. 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)
21. Leno, V., Polyvyanyy, A., Dumas, M., La Rosa, M., Maggi, F.M.: Robotic Process Mining: Vision and Challenges. Bus. & Inf. Sys. Eng. pp. 1–14 (2020)
22. de Leoni, M., Marrella, A.: Aligning Real Process Executions and Prescriptive Process Models through Automated Planning. Exp. Syst. with App. **82** (2017)
23. Liu, X.: Unraveling and Learning Workflow Models from Interleaved Event Logs. In: 2014 IEEE Int. Conf. on Web Services. pp. 193–200 (2014)
24. 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)
25. Marrella, A.: Automated Planning for Business Process Management. J. Data Semantic **8**(2), 79–98 (2019)
26. Marrella, A., Catarci, T.: Measuring the Learnability of Interactive Systems Using a Petri Net Based Approach. In: 2018 Conf. on Designing Interactive Systems. pp. 1309–1319 (2018)
27. 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)
28. Schmitz, M., Dietze, C., Czarnecki, C.: Enabling Digital Transformation through Robotic Process Automation at Deutsche Telekom. In: Digitalization Cases, How Organizations Rethink Their Business for the Digital Age, pp. 15–33 (2019)
29. 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)