A Method to Improve the Early Stages of the Robotic Process Automation Lifecycle

Andres Jimenez-Ramirez[1], Hajo A. Reijers[2], Irene Barba[1], and Carmelo Del Valle[1]

[1] Departamento de Lenguajes y Sistemas Informáticos, University of Seville, Seville, Spain
{ajramirez,irenebr,carmelo}@us.es
[2] Utrecht University, Utrecht, The Netherlands
h.a.reijers@uu.nl

Abstract. The robotic automation of processes is of much interest to organizations. A common use case is to automate the repetitive manual tasks (or processes) that are currently done by back-office staff through some information system (IS). The lifecycle of any Robotic Process Automation (RPA) project starts with the analysis of the process to automate. This is a very time-consuming phase, which in practical settings often relies on the study of process documentation. Such documentation is typically incomplete or inaccurate, e.g., some documented cases never occur, occurring cases are not documented, or documented cases differ from reality. To deploy robots in a production environment that are designed on such a shaky basis entails a high risk. This paper describes and evaluates a new proposal for the early stages of an RPA project: the analysis of a process and its subsequent design. The idea is to leverage the knowledge of back-office staff, which starts by monitoring them in a non-invasive manner. This is done through a screen-mouse-key-logger, i.e., a sequence of images, mouse actions, and key actions are stored along with their timestamps. The log which is obtained in this way is transformed into a UI log through image-analysis techniques (e.g., fingerprinting or OCR) and then transformed into a process model by the use of process discovery algorithms. We evaluated this method for two real-life, industrial cases. The evaluation shows clear and substantial benefits in terms of accuracy and speed. This paper presents the method, along with a number of limitations that need to be addressed such that it can be applied in wider contexts.

Keywords: Robotic process automation · Process discovery · Business process outsourcing

1 Introduction

The term Robotic Process Automation (RPA) refers to a software paradigm where robots are programs which mimic the behavior of human workers

interacting with information systems (ISs) and whose objective is to perform structured and repetitive tasks quickly and profitably [12,22,27]. Significant cost savings, agility, and quality improvement are associated to a successful RPA project [6]. Nonetheless, not all processes are suitable for automation. The following criteria need to be met [12]: the process (1) must be highly frequent, (2) with a low level of exceptions, (3) involving an enclosed cognitive scope, and (4) susceptible to human errors. According to these criteria, the best candidates to be subjected to RPA projects are processes found within the back-offices of a company [13].

In general, an RPA project follows the following lifecycle:

1. An analysis of the context to determine which processes – or parts of them – are candidates to be robotized, considering the criteria mentioned earlier.
2. The design of the selected processes, which involves the specification of the actions, data flow, etc., which must be developed.
3. The development of each designed process.
4. The deployment of the robots in their individual environments (e.g., virtual machines) to perform their jobs.
5. A testing or control phase in which the performance of each robot is analyzed and errors are detected. Noted that within the traditional software development lifecycle testing precedes deployment, RPA is characterized by lacking a testing environment; only the production environment is available.
6. The operation and maintenance of the process, which takes into account each robot's performance and error cases; the outcomes of this phase enable a new analysis & design cycle to enhance the robots.

When considering state-of-the-art RPA technology, specifically solutions like UIPath [25] or WorkFusion [29], it is apparent that these mostly focus on the later stages of the lifecycle, i.e, the actual development and deployment stages. They provide very limited support to detect candidates of tasks or processes to automate. To date, there are also no established methods to carry out these analysis and design phases. As a result, current RPA projects mainly rely on the analysis of documentation, which may be of poor quality and may require substantial effort to understand. Considering that designed robots are typically deployed in production environments, where they interact with operational ISs, there is much risk involved with building on an inaccurate analysis.

Interestingly, in companies that have adopted RPA, people and robots typically work side by side within their back-offices. People are in charge of managing the processes that are either not suitable for RPA or which still need to be robotized. To carry out their work, these human employees receive formal training on how to deal with the cases that belong to the process under their control, but also receive training-on-the-job: they experience exceptions, process deviations, and undocumented cases on a daily basis. We will leverage their domain knowledge and analyze their behavior as the main ingredients of a method that improves and speeds up the early stages of the RPA lifecycle.

Against this backdrop, we propose in this paper a method that starts by monitoring the computers of the back-office staff in a non-invasive manner by

recording the screen, mouse and key events. Next, we apply a set of mechanisms (e.g., image similarity and frequency analysis) to transform this information into a standardized event log [1] including UI information, i.e., the sequence of raw images is converted to a set of shorter sequences of events each one corresponding to a process instance that the back-office staff has performed. The generated log is then used to automatically discover the underlying process. As widely acknowledged [3,13,16], the process mining paradigm [2] provides efficient and suitable analytic techniques to address this step.

To evaluate our proposal, it has been applied to two industrial cases. We chose the domain of business process outsourcing (BPO) to select these cases since they provide a particularly challenging setting. When a process is outsourced, various companies are involved in carrying it out and the ISs in this process context may be geographically dispersed. What is more, the use of secured connections (e.g., Citrix) is the standard in outsourced scenarios, which only permits raw images to be gathered from the monitored screen instead of the structure of the information that is being processed. The results of our evaluation show clear benefits in the early phases of RPA projects in the BPO domain by (1) improving accuracy and (2) saving time. In addition, our evaluation shows that the insights generated during the analysis phase are also beneficial during further stages of the RPA lifecycle.

The rest of the paper is organized as follows. Section 2 motivates the need for leveraging human knowledge to foster better RPA analysis and design in a BPO context. Section 3 presents the details of our proposed method. Section 4 reports the results for the cases we used for our evaluation. Section 5 presents related work on RPA. Finally, Sect. 6 concludes the paper with a summary and description of future work.

2 Context

In a BPO scenario (cf. Fig. 1), back-office tasks are carried out both by teams of humans and teams of robots. For both types of teams, documentation of the prescribed process is a crucial ingredient. This documentation covers the various case situations in a mix of formal and informal descriptions, typically containing some sort of ambiguity or uncertainty (e.g., lacking information related to the software ecosystem beyond the ISs which are used). We will take a look at the role that documentation plays in more detail first.

As far as the human staff is concerned, documentation is used as the basis for training them. While human performers are expected to be able to deal with a particular process on the basis of this training, they are likely to encounter non-documented scenarios. Humans are expected to make decisions under such circumstances, which they are usually capable of by applying a cognitive effort – probably by applying common sense (i.e., *on-the-job training*). Typically, such new decision-making behavior is then incorporated into a knowledge database, which is shared across the team.

As explained earlier, the deployment of a robot team emerges from following the common RPA lifecycle, i.e., by going through the phases of analysis, design,

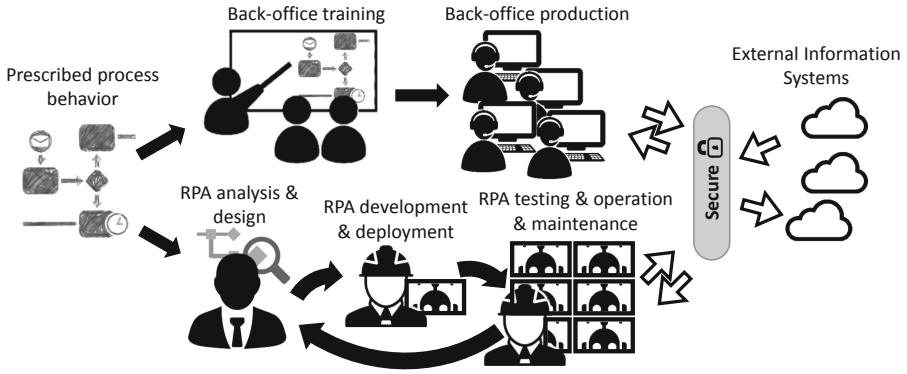

Fig. 1. Current situation of the business.

development, deployment, testing, operation, and maintenance. In the analysis and design phases, a significant effort is spent on formalizing the process to be carried out in such a way that ambiguity and uncertainty are reduced about the way various cases should be dealt with. Selected parts of the prescribed process are formally documented and delivered to the developers of the robots. These developers produce the robot code, which is then gradually deployed such that the robots can execute the related processes while their behavior is monitored and tested. After a period of time, a full team of robots will be ready to execute the process in question. However, robots tend to be rigid and do not behave well under unexpected circumstances. When an unexpected situation occurs, two things happen: (1) the involved robot gets stuck and a human operator must bring the process to a situation where the robot can go on – typically, the starting point of a new case, and (2) the RPA lifecycle starts again to analyze the newly encountered situations to decide which of these must be included in the design to deploy an updated version of the robot team.

From a company's perspective, each team – human or robotic – has its own benefits and drawbacks, which can be seen in Table 1. While robots can work 24/7 and can be flexibly spawned to deal with heavy workloads, they are expensive to analyze and have a low accuracy when dealing with judgment-based tasks, e.g., deciding what to do based on unstructured data. Therefore, they require the supervision of humans, which are typically engaged with knowledge-intensive tasks and relieved of repetitive, simple tasks [5].

As stated before, analyzing a process that is to be robotized is a time-consuming task and only a small portion of the process can be robotized initially, i.e., the most structured and repeated parts [3]. During the lifetime of such a process, new scenarios and corrections are detected and included in the robotized set. Eventually, only the most unstable and judgment-based parts of a process remain separated from the cases that can be handled by robots. The human team will stay in charge to deal with these. The sooner this eventual situation is reached, the more profitable the RPA project is.

Table 1. Main benefits and drawbacks of human vs robot taskforce which are identified in the considered scenario.

	Human	Robot
Benefits	Cognitive effort for problem solving	Continuous and flexible work capacity
Drawbacks	Not efficient for repetitive, high volume, simple tasks	Expensive to analyze and unreliable for problem solving

The context we provided here points at the main challenge that is being addressed in this paper: to reduce the effort to analyze the actual system. As the reader may have noticed, this effort is done twice: (1) when the human team is trained or *trained-on-the-job*, and (2) when new documentation is received or when errors are detected, which trigger a new analysis of the robots. Although managers of the human teams and managers that are concerned with the development of robots communicate with each other, the flow of knowledge that is available in the human teams to that of the development teams is far from ideal. To mend this, the current paper tries to automate this flow.

3 Method

In this section, we will describe our proposal to deal with the challenges that exist in an RPA setting. Essentially, this method leverages the knowledge of back-office staff for improving the early stages of the RPA lifecycle. This method starts with behavioral monitoring (cf. Sect. 3.1), after which a set of configuration cycles are conducted over the monitored information (cf. Sect. 3.2). Each cycle produces a process model that is related to such behavior (cf. Sect. 3.3). Figure 2 depicts an overview of this proposal.

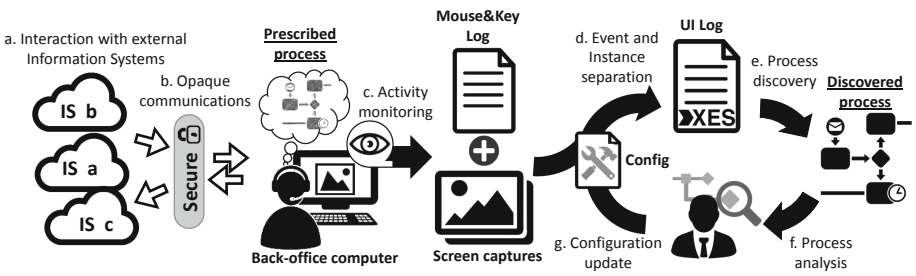

Fig. 2. Overview of the proposal.

3.1 Behavioral Monitoring

As can be seen in Fig. 2, in a BPO setting, a back-office employee interacts with different ISs (e.g., proprietary ERP systems cf. Fig. 2a) to perform a number of repetitive, administrative processes. The interaction with these ISs is done through secured connections (e.g., Citrix cf. Fig. 2b), which makes it difficult to perform a transparent analysis of the user interaction by means of tree-like structures (e.g., HTML DOM or Windows UI). Therefore, only raw images on the one hand and mouse and keyboard events on the other can be monitored from someone's computer during their interactions with an IS. Our method starts by recording these back-office interactions through non-intrusive monitoring software (i.e., it is invisible for the user, cf. Fig. 2c). This software captures a single long trace of events with the information depicted in Table 2.[1] Such events may be related to several traces (or instances) of the same process.

3.2 Configuration Cycles

A specific algorithm is executed to identify events and traces/instances to generate a UI Log (cf. Fig. 2d, Definition 1, Algorithm 1). For this step, a default configuration (i.e., a set of pairs $key-value$) is used with the information shown in Table 3.

Definition 1. *An **UI Log** is a XML file whose grammar is an extension of the standard XES [1]. This extension incorporates the* concept *standard [1] attributes (i.e.,* concept:name *for traces and events) and the attributes of Table 2.*

Algorithm 1. From raw log to UILog

input : Log l, Config c
output: UILog uiL

$uiL \leftarrow createBasicLog(l)$
$uiL.groupEventsByFingerprint(c.similarity\_th)$
$uiL.deleteEventsByTemplates(c.templates)$
$uiL.deleteEvents(c.rm\_events)$
foreach $(source, targets) : c.join\_events$ **do**
    $uiL.joinEvents(source, targets)$
if *c.st_event is not defined* **then**
    $c.st\_event \leftarrow uiL.mostReapeatedEvent()$
$uiL.divideLogInTracesByStartEvent(c.st\_event)$

[1] These event attributes are considered necessary for different use cases. However, not all of them are useful for the current paper.

Table 2. Event attributes captured from the UI.

Name	Description	Name	Description
app_name	Name of the application which is being used. Not useful in secured systems	*keystrokes*	Sequence of keystrokes pressed. Only if *event_type* is keystroke
focus	Indicates if the application has just gained the focus	*start_ts*	Start timestamp
event_type	$\{click, keystroke\}$	*end_ts*	End timestamp
click_type	$\{left, right, middle\}$. Only if *event_type* is click	*img_name*	Name of the file with the screen capture
click_coords	Coordinates of the click. Only if *event_type* is click	*img_fingerprint*	Quasi-unique string summarizing the image

Table 3. Configuration attributes.

Key	Description
similarity_th	Threshold to use for grouping images. If it increases, it will be more likely that two images are considered the same event
st_events	Set of event to be considered the alternative starts of the instances
rm_events	Set of events to be removed from the log
rm_templates	Set of images to remove the events which contains any of them
join_events	Set of pairs $(source, target)$ to join the source with the target events
sp_processes	Set of processes to be considered *special or exceptional* and, thus, to be separated from the final model

In a first step (cf. line 1 of Algorithm 1), the UI Log is created using the basic information of the input log i.e., the data of Table 2. Next, the images are used to identify repeated events (i.e., atomic process activities) and divide the long trace into different smaller traces.

First of all, repeated events are identified through an image-similarity algorithm which groups similar images (e.g., a log-in window with two different user names will be grouped together). In this step (cf. line 2 of Algorithm 1), the Hamming distance [14] of image fingerprints (e.g., pHash or Image-match [28]) is used to check whether two images correspond to the same activity or event,

i.e., if such a distance is below the configured threshold, then both images are grouped together. As a result, each event of the log now contains the event *concept* : *name*. As the log may contain noise, some mechanisms are enabled to clean the log from it (cf. lines 3–6 of Algorithm 1):

1. Some events can be deleted if they contain some configured templates (e.g., a social network or email icon, cf. Table 3 *rm_templates*).
2. Some detected events can be directly deleted (e.g., the log-in window of an ERP system, cf. Table 3 *rm_events*).
3. Some events can be considered to be the same although they look different (cf. Table 3 *join_events*).

Secondly, to separate the one-trace log into different traces, an event (i.e., the most repeated or a configured set, cf. lines 7–8 of Algorithm 1, Table 3 *st_events*) is used to split this long trace into smaller traces, each of which corresponds to a different instance (cf. line 9).[2] Therefore, in the resulting UI log, traces with different *concept* : *name* are created containing their corresponding events.

3.3 Process Discovery

The generated UI log is used as input of a process discovery algorithm (cf. Fig. 2e). The purpose of this step is to generate graphical process models that capture the behavior of the back-office employee. It is here that a business analyst becomes involved. First, the processes which meet a configured specification are marked as special (cf. Table 3 *sp_processes*). These relate to less relevant processes, which either obscure the main process or convert it into a spaghetti model. What is more, this mechanism can be used to exclude undesired processes, e.g., when the employee deals with multiple procedures on a daily basis. Second, the main process that is cleaned from the special cases is captured too.

Although the aforementioned steps (cf. Fig. 2d and e) are automatic, there are some parts where errors may be introduced. For example, the back-office employee may introduce noise in the UI log if she performs personal activities, the image-similarity algorithm may group images together that actually correspond to different events, and a wrong event may be selected to divide the cases. To deal with this, our proposal includes a manual activity by a business analyst to analyze the generated models (cf. Fig. 2f). In this step, the business analyst can perform a deep review of both the special model and the main model. In case she considers this necessary, any process model may be refined by providing a new configuration setting for a next iteration of the process discovery stage (cf. Fig. 2g). The process model that results from a number of iterations exactly captures how the back-office employee applies her know-how.

The outcomes of this approach can be used to improve different phases of the RPA lifecycle, as follows:

[2] Note that considering an event for dividing the traces implies that this selected event may not appear in the middle of a trace.

1. The final process model represents a key element for the analysis and design phases since it includes (1) information about the real process – which may differ from the documented one–, (2) figures on the frequency of the cases – which is relevant to decide which cases should be robotized–, and (3) the human effort that is still required to be allocated to the process – which is necessary to evaluate the potential performance of the RPA deployment.
2. The additional UI log information, which is associated to the activities of the process model, provides strong support for the development phase since it includes the actions that have been done (i.e., keystrokes and clicks) that lead one activity to other for the different instances of each case type. Furthermore, testing the developed robots will become easier since the UI logs can be used to generate test scenarios.

In the evaluation that follows, we will focus on our primary objective, the optimization of the analysis and design phases.

4 Evaluation

To evaluate our method, we worked together with Servinform S.A.[3], which is active in the BPO domain. To be more precise, we carried out the analysis and design phases for RPA scenarios within two companies. We compared the outcomes with those that resulted from the conventional analysis and design activities of Servinform for the same scenarios. Section 4.1 describes the set-up that underlies our evaluation; Sect. 4.2 provides the data analysis and summarizes the results of the evaluation.

4.1 Set-Up

To enable our evaluation, a software infrastructure is developed and deployed (cf. Fig. 3) to support the following use case:

1. A back-office computer is monitored for some time through non-intrusive software written in the AutoIT scripting language [7]. The software sends all events (cf. Table 2) to a central server using JSON messages.
2. A central server, coded using Java and the Spring Cloud framework [23], exposes a REST API for receiving the events of the monitored computer and storing both the images and the UI events into a MySQL database. The fingerprint of the images is calculated by a Python component[4].
3. After the observation period, an analyst requests a process model that is based on the monitored events to the central server. For this purpose, a configuration (cf. Table 3) is sent through a custom-made ProM plugin.

[3] Serviform is a Spanish BPO company with an IT consulting area.
[4] There are several alternatives for computing a fingerprint of an image, in this paper we based on [28].

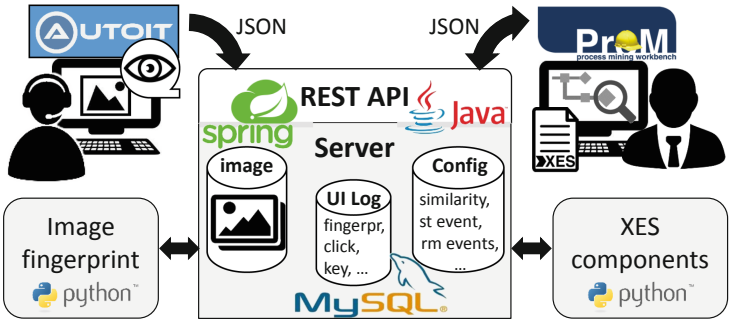

Fig. 3. Developed infrastructure.

4. The central server stores the configuration and applies it to the collected event log using specific components written in Python. Such components take care of grouping events, separating instances, etc. to eventually generate a XES document (cf. Definition 1) which is sent back to the ProM plugin.
5. The analyst could in principle select any process mining algorithm that is available in the ProM interface to discover the process model behind the XES document. Nonetheless, due to this special evaluation context, we developed a simple algorithm based on [21], which generates EPC models [4] by merging the process instances of the log. Such models are shown in the ProM interface (cf. Fig. 4) to be reviewed by the analyst. The analyst may either decide to finish the discovery cycle or generate a new configuration, thus going back to step 3 of the use case.

This use case has been applied to two outsourced processes, which are carried out by two different companies. These companies, a major Spanish bank and a telecommunication company, consented to cooperate for the sake of this evaluation, but did not authorize the disclosure of their names. Therefore, we will refer to them as *company*1 and *company*2. The processes of interest are well known by Servinform and have been in production for months. For the purpose of this evaluation, we obtained the results from an *a priori* analysis and design for each process, as carried out by Servinform. We will refer to the models that describe the behavior of the two processes as the *a priori* models. These models provide us with the insight into the understanding of the processes on the basis of a conventional, document-oriented analysis and design phases. The *a priori* model for *company*1 contains 5 alternative paths (i.e., different sequences of process activities); for *company*2, there are 9 different paths. Note that the relative simplicity of the complexity of these models is precisely the reason that they were considered as candidates for RPA.

Our method has been deployed, which means in particular that our software was installed and was executed on two back-office computers, one for each company. The software has been actively gathering and analyzing events for a period of one week. In general, an appropriate logging involves a trade-off between reach-

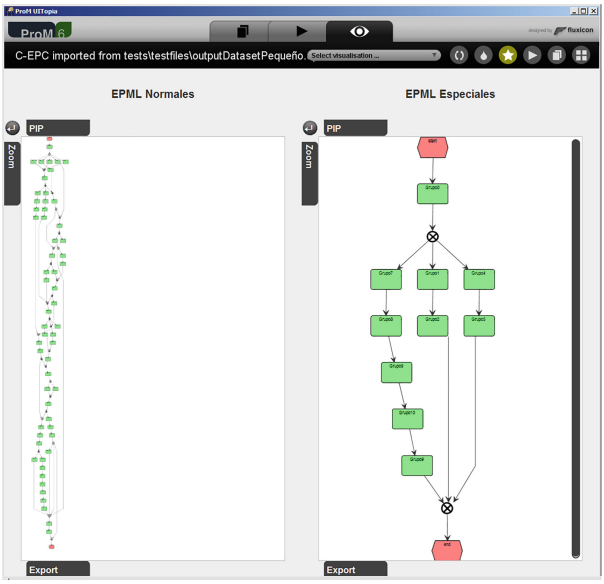

Fig. 4. Main (left) and special (right) processes visualization in the ProM plugin.

ing a proper size of the log and working towards a reasonable time to start the process analysis. We considered one week to be appropriate since it guaranteed us to observe, at least, 200 completely logged cases for both processes. This amount of cases for processes with an a priori complexity of approximately 10 alternative paths seemed adequate. After one week of observation, a business analyst with general knowledge of the underlying, outsourced processes started working with the ProM plugin to go through different configuration cycles for each process. For the sake of explaining our evaluation results, we will only consider (1) the *initial model* (i.e., the first model that is automatically created on the basis of a default configuration, cf. Sect. 3.2), and (2) the *final model* (i.e., the last model that is discovered after going through some configuration cycles and a considerable cognitive effort). These two models are stored with the goal of being analyzed by the company staff.

To compare for each process the *a priori* model with the discovered models, we used different measures:

1. *#paths a priori*, *#paths initial* and *#paths final*: the number of process paths which are included in the *a priori* model, initial model, and final model respectively,
2. *%new*: the percentage of paths that are present in the final model but cannot be found in the *a priori* model,
3. *%non-discovered*: the percentage of paths that are present in the *a priori* model that cannot be found in the final model.

4.2 Results and Conclusions

Table 4 shows the values that we obtained for the measures we introduced. As can be seen from this table when considering the *#paths a priori* and *#paths final* columns, both scenarios show that our method discovers a greater variety of paths than the conventional approach. This difference is mainly due to the fact that new paths are discovered. Specifically, in both cases more than 30% of the paths within the final models are new. This means that the monitored employees have dealt with cases in ways that were not *a priori* modelled yet exist in real life. When considering the *non-discovered* column, it is interesting to note that the final model totally captures the paths in the *a priori* model for the process within the first company, but that there is one path out of the 9 in total in the *a priori* model for the second company (11%) that is not found back in the final model. This can be interpreted in different ways. It could hint at the event log to be too small (and the observation period to be too short) to identify all paths. However, given the low complexity of the path in question, it could also be seen as an indication that this particular path is actually not considered so relevant by the business analyst.

Table 4. Result regarding the *a priori* and the final models.

Company	#Paths a priori	#Paths initial	#Paths final	%New	%Non-discovered
#1	5	46	12	7/12 (**58%**)	0/5 (**0%**)
#2	9	60	13	5/13 (**38%**)	1/9 (**11%**)

It is also interesting to briefly review the *#paths initial* column, i.e. the ones that are generated on the basis of default configuration settings. When analyzing these models, many more paths can be seen – 46 in *company*1 and 60 in *company*2. Yet, many of them are of poor quality since they are based on noise and irrelevant events. This gives us the insight that it is essential to combine the automatic steps of our proposal with the cognitive effort of the business analyst: this combination transforms the big sets of *bad* paths within the initial models into smaller numbers of *high quality* paths within the final models.

Finally, we also present here some additional insights that we collected from the feedback from the business analyst who was involved in this evaluation on the overall method:

1. The increase in the covered process behavior in the final models is significant and relevant. Moreover, some of the new paths may not belong only to the outsourced processes (i.e., the interaction with the external ISs) but to company's own internal computer ecosystem, e.g., the process to solve a common network connection problem which is not related to the outsourced process. Therefore, the proposal may also detect paths that are unlikely to appear in any *a priori* model based on documentation.

2. The tools that are part of our method appear useful to clean the initial models from noise and to decide on which paths are relevant. When it happens that some relevant paths correspond to small repetitive patterns, like exceptions that may happen in different longer paths (e.g., closing an unexpected pop-up and then continue with the normal path), it does mean that the analyst must decide between throwing out the full path or making a stronger cognitive effort to separate such patterns and capture this knowledge.
3. The final models include additional, relevant data about process paths when compared with the *a priori* models (e.g., path frequency, mouse actions, time, etc.). In our evaluation, we only focused on the use of the frequency of paths since this was acknowledged to be most relevant for the analyst.
4. Although time has not been measured in this evaluation, discovering the final model by the analyst turned out to be a matter of hours, i.e., the time between the first configuration is done until the final process model is generated. To compare: in the opinion of the analyst, the conventional, document-based analysis and design of the *a priori* model is a matter of days or even weeks. This provides preliminary indications that this proposed method can indeed help to also speed up the analysis and design phases.

In summary, this evaluation clearly shows the improved accuracy of analyzing processes using our tool-supported method in comparison with extracting knowledge process knowledge from documents. We also obtained some tentative insights on additional benefits, most importantly about shortening the time required to discover the relevant parts of processes that can be robotized.

5 Related Work

In recent years, other authors have also proposed techniques that can be applied in the early stages of an RPA project. Specifically, [15] summarizes a number of best practices and provides guidance to prioritize processes for the analysis phase. However, this work lacks any technical support for this phase. From a more technical point of view, [17] proposes the use of natural language processing and machine learning for detecting candidate activities from a textual description of processes. Unlike [17], our proposal analyzes the actual behavior of the system instead of what is available in the documentation. In that sense, we provide a new perspective of relevant information that seems worth considering. In turn, the authors of [18] introduce the concept of *desktop activity mining*. Similar to our method, this work combines monitoring techniques with process mining [2]. However, it would not be feasible to apply this in complex settings such as the BPO domain we considered since it requires access to actual UI elements. Similarly, [16] suggests that using process mining for discovering local process models (i.e., frequent patterns) from UI logs may be useful to train robots. Also, [3] describes how RPA may leverage techniques from process mining paradigm. Unlike our proposal, [3,16] focus on a characterization of the problems involved and only suggest in abstract terms how they can be addressed.

There are other research areas related to our approach. On the one hand, the human-computer interaction community also faces the problem of understanding user behavior using logs [11]. Instead of discovering the underlying process, [19] proposes a method to check the alignment between the user log and a Petri Net which model the expected interaction of the user. In turn, [10] analyzes event logs to group events in frequent and meaningful tasks. Similarly, [26] proposes a grouping approach to differentiate the kind of users that interact with an interface. However, [10,26] rely on a previous identification of click operations while our approach is based on low-level clickstreams.

On the other hand, the process mining community pays special attention to event log preprocessing for enhancing log quality. It is recognized that real world logs are often noisy because some of their traces are duplicated, incomplete, inconsistent, or reflect some other incorrect behavior [8]. The presence of noise in the event log leads to unnecessarily complex discovered models that do not accurately reflect the underlying process [24]. Within the process mining community, various techniques have been proposed to discover and remove noise (e.g., [9,20]) that can be adapted to be a part of the cleaning mechanisms that are part of our approach.

6 Conclusion

This paper presents a method for improving the analysis and design phases of the RPA lifecycle in a BPO context. This context is specially challenging since, in general, it implies using secured connections in a way that the ISs to interact with are seen as black box systems: the inputs are mouse and keyboard events and the output is a raw image related to the screen of the ISs.

To address such a complex scenario, we described our main idea: to leverage the knowledge of back-office staff who interact with the ISs on a daily basis. To do so, their computers are monitored and this information is then transformed into a UI log using a series of image-analysis algorithms. Next, this UI log is refined and used to discover the underlying process model using techniques from the process mining paradigm.

This paper describes the steps and the tools that are part of the proposed method. It has been applied to two real-life processes, which form the basis of our evaluation. The results indicate that the proposed method is suitable for the considered problems. Principally, it improves the accuracy of the process analysis. Among the additional benefits, there are indications that this method would considerably speed up the early stages of an RPA analysis as well.

Nonetheless, the approach presents some limitations that are planned to be mitigated as future work. First, the proposed method considers a log regarding a single-user interaction. In case that multiple back-office computers are monitored, a variety of solutions may exist, but each of these need to be evaluated. For instance, a procedure could be to discover each single-user process model first and then merge them into a single process model. Second, this paper generates a process model that mimics the behavior of a user including possible

errors or inefficiencies. Although some of these may be identified and cleaned from the log by using our approach, others may require further improvements and optimizations that are out of the current scope. Third, the behavior which is analyzed only covers the events which are observable, i.e., performed through the back-office computer. Identifying such non-observable events is useful since the existence of them in a path indicates that such a path is not a good candidate to be robotized.

As further future work, we plan (1) to investigate how further phases can be enhanced on the basis of logged events and automated discovery techniques, (2) to evaluate the current method in a controlled experiment that, on the one hand, considers experts' opinions regarding the paths discovered and, on the other hand, focuses on measuring time gains since the current conclusions about time may not be generalizable to other scenarios, (3) to analyze the impact of using different process discovery algorithms for generating the final model, and (4) to investigate further mechanisms to clean and enhance the UI log, e.g., identifying activities using screen-scraping algorithms.

Acknowledgments. This research has been supported by the Pololas project (TIN2016-76956-C3-2-R) of the Spanish Ministerio de Economía y Competitividad. Special thanks to Rafael Cabello from Serviform S.A. for providing his invaluable support and access to the case data.

References

1. IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849–2016, pp. 1–50 (2016)
2. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
3. van der Aalst, W.M.P., Bichler, M., Heinzl, A.: Robotic process automation. Bus. Inf. Syst. Eng. **60**(4), 269–272 (2018)
4. van der Aalst, W.: Formalization and verification of event-driven process chains. Inf. Softw. Technol. **41**(10), 639–650 (1999)
5. Aguirre, S., Rodriguez, A.: Automation of a business process using robotic process automation (RPA): a case study. In: Figueroa-García, J.C., López-Santana, E.R., Villa-Ramírez, J.L., Ferro-Escobar, R. (eds.) WEA 2017. CCIS, vol. 742, pp. 65–71. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66963-2_7
6. Asatiani, A., Penttinen, E.: Turning robotic process automation into commercial success - case opuscapita. J. Inf. Technol. Teach. Cases **6**(2), 67–74 (2016)
7. Autoit (2018). https://www.autoitscript.com/site/autoit/. Accessed 1 Mar 2019
8. Cheng, H.J., Kumar, A.: Process mining on noisy logs - can log sanitization help to improve performance? Decis. Support. Syst. **79**, 138–149 (2015)
9. Conforti, R., La Rosa, M., ter Hofstede, A.H.: Noise filtering of process execution logs based on outliers detection (2015)
10. Dev, H., Liu, Z.: Identifying frequent user tasks from application logs. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces, pp. 263–273 (2017)

11. Dumais, S., Jeffries, R., Russell, D.M., Tang, D., Teevan, J.: Understanding user behavior through log data and analysis. In: Olson, J.S., Kellogg, W.A. (eds.) Ways of Knowing in HCI, pp. 349–372. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-0378-8_14
12. Fung, H.P.: Criteria, use cases and effects of information technology process automation (ITPA). Adv. Robot. Autom. **3**(3), 1–10 (2014)
13. Geyer-Klingeberg, J., Nakladal, J., Baldauf, F., Veit, F.: Process mining and robotic process automation: a perfect match. In: International Conference on Business Process Management, pp. 1–8 (2018)
14. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1999)
15. Le Clair, C.: Digitization Leaders Share Robotic Process Automation Best Practices. Forrester Research Inc., Cambridge (2016)
16. Leno, V., Dumas, M., Maggi, F.M., La Rosa, M.: Multi-perspective process model discovery for robotic process automation. CEUR Work. Proc. **2114**, 37–45 (2018)
17. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 67–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91704-7_5
18. Linn, C., Zimmermann, P., Werth, D.: Desktop activity mining - a new level of detail in mining business processes. In: Workshops der INFORMATIK 2018 - Architekturen, Prozesse, Sicherheit und Nachhaltigkeit, pp. 245–258 (2018)
19. Marrella, A., Catarci, T.: Measuring the learnability of interactive systems using a petri net based approach. In: Proceedings of the 2018 Designing Interactive Systems Conference, pp. 1309–1319. ACM (2018)
20. 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 Min. Knowl. Discov. **13**(1), 67–87 (2006)
21. Rosa, M.L., Dumas, M., Uba, R., Dijkman, R.M.: Business process model merging: an approach to business process consolidation. ACM Trans. Softw. Eng. Methodol. (TOSEM) **22**, 11 (2012)
22. Slaby, J.R.: Robotic automation emerges as a threat to traditional low-cost outsourcing. Horses for Sources (2018)
23. Spring cloud (2018). https://spring.io/projects/spring-cloud. Accessed 1 Mar 2019
24. Suriadi, S., Andrews, R., ter Hofstede, A.H., Wynn, M.T.: Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. Inf. Syst. **64**, 132–150 (2017)
25. Uipath (2018). http://www.uipath.com. Accessed 1 Mar 2019
26. Wang, G., Zhang, X., Tang, S., Zheng, H., Zhao, B.Y.: Unsupervised clickstream clustering for user behavior analysis. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 225–236. ACM (2016)
27. Willcocks, L., Lacity, M., Craig, A.: Robotic process automation: strategic transformation lever for global business services? J. Inf. Technol. Teach. Cases **7**(1), 17–28 (2017)
28. Wong, C., Bern, M.W., Goldberg, D.: An image signature for any kind of image. In: International Conference on Image Processing, pp. 409–412 (2002)
29. Workfusion (2018). http://www.workfusion.com. Accessed 1 Mar 2019