A Tool for Incorporating Eye Tracking Data in RPA: Enhancing User Behavior Logs

M. García-Romero[1], Antonio Martínez-Rojas[1], J. G. Enríquez[1] and A. Jiménez-Ramírez[1]

[1] *University of Seville. Department of Computer Languages and Systems. E.T.S. Ingeniería Informática. Avd. Reina Mercedes, s/n, 41012, Seville, Spain*

Abstract

This paper presents UBGI, an innovative tool designed to enhance Robotic Process Automation (RPA) by integrating eye tracking data with user interface (UI) logs. UBGI processes and combines gaze logs with UI logs to create enriched User Behaviour (UB) logs, enabling more precise identification of user focus areas. By applying filtering masks to screenshots, UBGI highlights relevant data, facilitating analysis of user interactions. This tool enables further analysis of user behaviour through an external source, specifically eye tracking data.

Keywords

Robotic Process Automation, Robotic Process Mining, User Behaviour Mining, User Behaviour Log, Eye Tracking

1. Introduction

In the current era of Hyperautomation [1], companies are driven to quickly identify and automate every possible business process. This trend coincides with the growing popularity of RPA. Unlike traditional automation methods, such as those based on APIs, RPA is based on graphical user interfaces to automate and integrate systems [2].

As the first condition for automating a process, it is necessary to understand how it is performed and even discover what process needs to be automated. To address this, the so-called mining techniques emerged. Mining techniques like Robotic Process Mining (RPM) [2] and User Behaviour Mining (UBM) [3] are essential to identify process models that represent human behaviour, facilitating automation of processes.

These techniques rely on the UI log [4], which records user actions as clicks or keystrokes within a graphical user interface. In addition, the information on the screen can be incorporated as features to provide context for user actions. This can be useful for analysing the routines or decisions recorded in the UI log [5].

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024.

mgarcia44@us.es (M. García-Romero); amrojas@us.es (A. Martínez-Rojas); jgenriquez@us.es (J. G. Enríquez); ajramirez@us.es (A. Jiménez-Ramírez)

0000-0003-2113-3497 (M. García-Romero); 0000-0002-2782-9893 (A. Martínez-Rojas); 0000-0002-2631-5890 (J. G. Enríquez); 0000-0001-8657-992X (A. Jiménez-Ramírez)

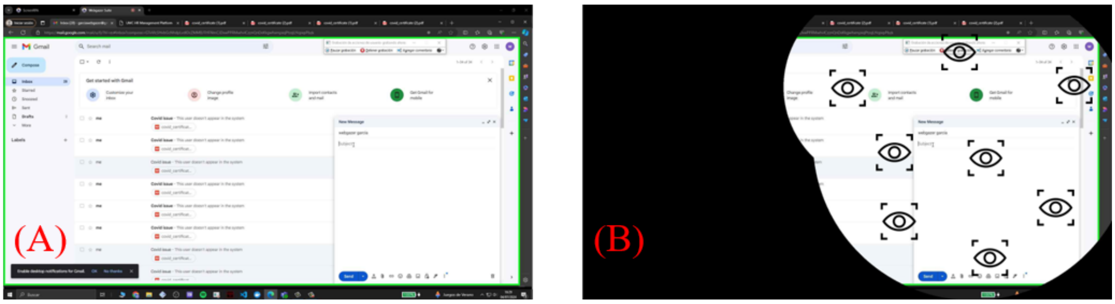

Figure 1: (A). A screenshot extracted from a UI Log. All UI element in screenshot must be processed later. (B). A screenshot obtained after executing UBGI. A *black gaze mask* hides all the regions where the user has not focused his gaze.

Nevertheless, a challenge arises because of the extensive amount of data contained in the screenshots. These screenshots capture relevant and irrelevant information, encompassing all UI elements displayed on the screen [6]. Consequently, extracting meaningful insights from UI logs becomes challenging, especially when dealing with complex and information-dense graphical user interfaces. This mixture of data requires careful filtering to ensure optimal performance. To address this problem, eye tracking technology can be applied to capture the points of the screen where the user focuses his gaze while interacting with the user interface [7]. Additional gaze information such as the Point of Gaze (POG), gaze fixations and dispersion [8] are obtained by eye tracking software based on webcams or infrared eye tracking tools. This additional gaze information is essential to define attention areas and enables us to distinguish between relevant and irrelevant data in the screenshots [6], as shown in Figure 1. These partially-masked screenshots can help by revealing the UI elements users need to interact with, connecting clicks or keystrokes to specific on-screen content.

This article introduces *User Behaviour and Gaze Integrator* (UBGI)[1], a tool capable of performing the following functions:

- Proccess and combine the eye tracking data extracted from the eye tracking software as a gaze log with the UI log defined in. The gaze log is any CSV file that includes gaze events as defined in [6]. The UI log supports formats such as XES/CSV/MHT that include user actions, as reflected in [9]. As a result of this combination, the User Behaviour (UB) log is obtained [7] .
- Generate screenshots from those obtained in the UI log with an overlay mask. This mask covers all regions of the screenshot where fixations have not been detected, leaving visible only the regions where fixations have been already detected. These visible regions in the screenshots represent the attention areas [6].

The UB log and the generated screenshots can be used as seen in [6] to filter information extracted from the screen to improve performance, which is useful for RPM approaches such as routine discovery proposals [4] or decision models [5].

The remainder of this work is structured as follows. Section 2 describes the main features of the tool. In Section 3, the maturity of the tool is discussed. Section 4 provides links to the demonstration of the tool in the form of a video. Finally, the conclusions of this work are presented in Section 5

[1] Source Code: https://github.com/RPA-US/ubgi

2. UBGI Tool Features

The proposed tool has been developed as a web application using *Django* REST framework for Python. JavaScript is used for client-side rendering, and PostgreSQL is used as the database.

The innovative features of the tool reside in two phases: the *UB Log Gaze Enricher* phase and the *Filtering Mask* phase. Before these two phases, the user must record the execution of the process, i.e., the case study. The case study consists of one or more UI logs and gaze logs for one or more scenarios. To obtain the UI log for each scenario with the screenshots associated with each action, loggers are used to monitor the user's activity and save it. To obtain the gaze log, it is necessary to have an eye tracking tool, such as webcam-based eye tracking software that can capture the POGs on the screen during the case study recording. WebgaWzer.js is an open source webcam-based eye tracking web application [2] [10]. It is already included in the UBGI tool.

Once the case study is available and loaded into the UBGI tool, it is executed through the following phases:

- **UB Log Gaze Enricher phase**: In this phase, we integrate the UI log with the gaze log to form the UB log. The eye tracking software data is analyzed to identify gaze events such as fixations, saccades, and dispersion values [8]. The integration of information from two different logs into a single log is made possible by timestamps (in UTC format), which allow us to match user actions, such as clicks and keystrokes, with the corresponding gaze events occurring within the time interval of a specific action and the next one. In this way, we obtain the UB Log enriched with gaze data.
- **Filtering Mask phase**: From the screenshots extracted from the UI log, a copy of each is generated. For each of these new screenshot copies, a *gaze-based mask* is created. This *gaze-based mask* consists of a black filter mask that covers the entire screenshot, except for the regions identified as attention areas [6]. The attention areas, which are the regions that are not overlapped by the filtering mask, are circles drawn from the centroid of each fixation with a radius determined by the value of the dispersion. The user can increase or decrease this parameter within the UBGI tool [6]. In this way, it is possible to observe which regions of the screen have been viewed by the user and which have not, to determine the relevant and irrelevant information between the user actions. Figure 1 shows an example of a screenshot copy generated in this phase.

The UB log generated in the *UB Log Gaze Enricher* phase and the screenshots that include the filtering mask can be downloaded as a.zip file once the case study is executed.

3. Tool Maturity

The UBGI tool is currently in a stable version that meets its purposes. Nevertheless, there are several lines of research and future work that can contribute to a higher level of maturity.

First, the UBGI tool demonstrates flexibility in adjusting different parameters and formats of UI logs. However, it does not have the same level of adaptability for gaze logs. In future work,

[2] Webgazer.js source code: https://github.com/brownhci/WebGazer.

it is necessary to deepen the understanding of the various available eye tracking software and apply a similar treatment to that used for UI logs. This will allow UBGI to process any gaze log from any eye tracking software.

Second, determining the attention areas and gaze masks in the *Filtering mask* phase only considers fixation and dispersion gaze events. Other parameters can be configured to determine an optimal drawing of the attention areas and, conversely, the filtering mask (e.g., thresholds, saccade eye movements, etc.).

Third, the artefacts generated in the phases of the UBGI tool contain important information only if they are going to be processed. Therefore, it is important to integrate the artefacts generated in the UBGI phases into new phases or modules that can be created within the tool for the discovery of automatable processes. Even include the UBGI tool in other tools that perform the function of extracting features from the UI for process discovery.

UBGI has been subjected to pilot tests in simulated environments that replicate real-use conditions. These tests have provided valuable data to identify possible improvements and confirm the effectiveness of UBGI in data integration and analysis. However, implementation in real productive environments will be crucial to evaluate its performance in intensive use situations and its capacity to handle massive amounts of data.

4. Tool Demonstration

In this section, we present a demonstration of the artefacts generated by the UBGI tool. To do this, we will record and execute a complete case study[3]. The case study simulates a process within the HR department of a company. Specifically, the case study involves an administrative worker who must attach the COVID 19 certificates of company employees to the database from a folder. If an employee is not found in the database, the administrator must report the missing employee with his COVID 19 certificate.

For recording the UI log, we use Steprecorders, a native application of the Windows operating system. To record the gaze log, we use the Webgazer.js eye tracking software.

5. Conclusions

The User Behaviour Gaze Integrator (UBGI) tool contributes to the BPM community with the ability to integrate and analyse eye tracking data and user interface logs, enabling the consideration of broader contextual information when studying user behaviour. Therefore, this opens up new lines of work in fields such as RPA, UBM, and RPM.

The proposed tool processes user activities by first integrating UI logs with gaze logs to form an enriched UB log. This is achieved through the UB Log Gaze Enricher phase, which matches user actions with corresponding gaze events using timestamps. In the subsequent Filtering Mask phase, the tool generates modified screenshots highlighting attention areas by overlapping a gaze-based mask, revealing relevant and irrelevant information found in the screenshots, whether the user is performing an action or not. These capabilities allow for a

[3] Video: https://youtu.be/hW4-oQRNBC8

more detailed analysis of user interactions, providing valuable insights to further analysis for process automation strategies.

Acknowledgments

This research was supported by the EQUAVEL project PID2022-137646OB-C31, funded by MICIU/AEI/10.13039/501100011033 and by FEDER, UE; the DISCOVERY project (2021/C005/00148631), funded by Unión Europea NextGeneration EU and “Plan de Recuperación, Transformación y Resiliencia” of the Ministry of Economic and Digital Transformation; and the grant FPU20/05984 funded by MICIU/AEI/10.13039/501100011033 and by FSE+.

References

[1] A. Jiménez-Ramírez, Humans, processes and robots: a journey to hyperautomation, in: International Conference on Business Process Management, Springer, 2021, pp. 3–6.

[2] W. M. Van der Aalst, M. Bichler, A. Heinzl, Robotic process automation, 2018.

[3] J.-R. Rehse, L. Abb, G. Berg, C. Bormann, T. Kampik, C. Warmuth, User behavior mining: A research agenda, Business & Information Systems Engineering (2024) 1–18.

[4] S. Agostinelli, M. Lupia, A. Marrella, M. Mecella, Reactive synthesis of software robots in rpa from user interface logs, Computers in Industry 142 (2022) 103721.

[5] A. Martínez-Rojas, A. Jiménez-Ramírez, J. G. Enríquez, H. A. Reijers, A screenshot-based task mining framework for disclosing the drivers behind variable human actions, Information Systems 121 (2024) 102340.

[6] A. Martínez-Rojas, H. A. Reijers, A. Jiménez-Ramírez, J. G. Enríquez, What are you gazing at? an approach to use eye-tracking for robotic process automation, in: International Conference on Business Process Management, Springer, 2023, pp. 120–134.

[7] A. Martínez-Rojas., A. Jiménez-Ramírez., J. G. Enríquez., D. Lizcano-Casas., Incorporating the user attention in user interface logs, in: Proceedings of the 18th International Conference on Web Information Systems and Technologies - WEBIST„ INSTICC, SciTePress, 2022, pp. 415–421. doi:10.5220/0011568000003318.

[8] D. D. Salvucci, J. H. Goldberg, Identifying fixations and saccades in eye-tracking protocols, in: Proceedings of the 2000 symposium on Eye tracking research & applications, 2000, pp. 71–78.

[9] A. Martínez Rojas, A. Jiménez Ramírez, J. Gonzalez Enriquez, H. A. Reijers, A tool-supported method to generate user interface logs, in: 56th Hawaii International Conference on System Sciences (2023), pp. 5472-5481., HICSS: Hawaii International Conference of System Sciences, 2023.

[10] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, J. Hays, Webgazer: scalable webcam eye tracking using user interactions, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, AAAI Press, 2016, p. 3839–3845.