A Reference Data Model
for Process-Related User Interaction Logs

Luka Abb[(✉)] and Jana-Rebecca Rehse

University of Mannheim, Mannheim, Germany
{luka.abb,rehse}@uni-mannheim.de

Abstract. User interaction (UI) logs are high-resolution event logs that record low-level activities performed by a user during the execution of a task in an information system. Each event in a UI log corresponds to a single interaction between the user and the interface, such as clicking a button or entering a string into a text field. UI logs are used for purposes like task mining or robotic process automation (RPA), but each study and tool relies on a different conceptualization and implementation of the elements and attributes that constitute user interactions. This lack of standardization makes it difficult to integrate UI logs from different sources and to combine tools for UI data collection with downstream analytics or automation solutions. To address this, we propose a universally applicable reference data model for process-related UI logs. Based on a review of scientific literature and industry solutions, this model includes the core attributes of UI logs, but remains flexible with regard to the scope, level of abstraction, and case notion. We provide an implementation of the model as an extension to the XES interchange standard for event logs and demonstrate its practical applicability in a real-life RPA scenario.

Keywords: User behavior mining · UI Log · Data model · Robotic process automation · Task mining

1 Introduction

User interaction (UI) logs are high-resolution event logs that record low-level, manual activities performed by a user during the execution of a task in an information system (IS) [1]. Each event in a UI log corresponds to a single interaction between the user and the graphical user interface (GUI) of a software application. Examples include clicking a button, entering a string into a text field, ticking a checkbox, or selecting an item from a dropdown [26]. Multiple recent research streams use this type of data, for example to analyze usage patterns in software applications [6,12,28], to identify candidate routines for robotic process automation (RPA) [7,26,35], or to derive RPA automation and test scripts [3,8]. In addition, companies like Celonis and UiPath offer tools that record and process UI data for inspecting and automating task executions [5].

The UI logs currently used in research differ substantially. The data collected in a specific research context is usually limited in scope and tailored to the proposed analysis technique or automation approach. This results in considerable variation regarding the number, type, and granularity of recorded events and corresponding attributes. Even when researchers record the same attributes at a similar level of detail, there is no common definition of UI log attributes to which they can adhere. Instead, they often rely on ad-hoc conceptualizations of elementary notions like activities and UI components. The situation is similar in industry, where each vendor has developed their own UI log format tailored to the capabilities of their recording software [27].

This lack of standardization makes it difficult to integrate UI logs from different sources [26,30]. It also poses a challenge for the interoperability of data collection and downstream processing tools: logs recorded by one tool are usually only compatible with the associated analytics or automation approach. Combining data collection and processing tools requires considerable preprocessing effort or is entirely infeasible if the necessary attributes cannot be recorded [27].

In this paper, we address these challenges by proposing a reference data model for process-related UI logs. This model provides a data structure and an accompanying interchange format that others can reuse to conceptualize and capture UI logs in a process context. To ensure widespread applicability, the model is designed such that it subsumes and integrates the commonalities of existing process-related UI logs, but remains flexible with regard to the their differences. To identify those commonalities and differences, we conduct a literature review in Sect. 3 and a review of industry solutions in Sect. 4. The reference data model, along with its underlying design principles and an accompanying interchange format, is presented in Sect. 5. In Sect. 6, we demonstrate how the data model can be instantiated in practice by applying it in a real-life RPA scenario. Finally, we conclude the paper with a discussion in Sect. 7.

2   Background and Related Work

Event Logs. Process mining extracts information from *event logs*, i.e., collections of *events* recorded in an IS [32]. An event log consists of *cases* that each correspond to one process instance. Each case contains a trace of events that occurred during the execution of the process instance and can have additional attributes, for example, the size of an order in an order-to-cash process. Events are related to a particular step in a process with an *activity* label (e.g., create invoice) and can also have additional attributes.

Data Formats. To enable the exchange of event data between different ISs, the business process management (BPM) community has developed interchange formats that define the structure and general contents of event logs. The current main format is XES (eXtensible Event Stream), which was introduced in 2010 to replace the older MXML format and was accepted as the official IEEE standard for event data in 2016 [38]. In XES, an event log consists of a three-level hierarchy of log, trace, and event objects. The format is designed to be highly generic,

with a minimal set of explicitly defined attributes on each of the three levels. Additional attributes, with a commonly understood semantic meaning, can be introduced by XES extensions. For example, the concept extension introduces the "name" attribute, which stores names for event logs, traces, and events. Although researchers have recently pointed out shortcomings of XES and proposed more flexible, object-centric alternatives such as OCEL [17], XES remains the most common event log format and is supported by many process mining tools.

UI Logs. UI logs are a particular type of event log in which events correspond to low-level interactions of a user with a GUI. They can be recorded either internally by adding logging capabilities to an application, or externally by dedicated logging tools. These tools record screen coordinates for each action and map them to parts of the GUI using optical character recognition technology.

User Behavior Mining. UI logs essentially record how users behave while they are engaged with an application. They can be analyzed by means of data or process mining techniques to gain data-driven insights into user behavior. We refer to this analysis of UI logs as *user behavior mining (UBM)* [1]. UBM can serve different purposes, including the analysis of software usage patterns, the design of new user assistance components, or the automation of tasks.

Task Mining. One application of UBM is to enhance traditional process mining by providing a more detailed view of execution steps. Event logs gathered from ERP systems like SAP or Oracle capture the main tasks in a process, like creating an order, but they do not provide insights into how employees actually perform these tasks. Recording and analyzing detailed task executions is referred to as *task mining* [32] or *desktop activity mining* [29]. These techniques can give companies deeper insights into their processes than traditional process mining alone, and they can also help software vendors to optimize their products, for example, by identifying common usability issues.

Robotic Process Automation. UI logs can be used to automate tasks and entire processes by having bots emulate the recorded user interactions. This approach to automation is called *robotic process automation (RPA)* [22] and has lately received considerable attention in research and practice. Within RPA, UBM techniques can be used to derive automation scripts, but also for *robotic process mining* [26], which for example encompasses the identification of suitable tasks for automation from UI logs [24].

Web Usage Mining. Another field that is concerned with the analysis of user behavior is *web usage mining* [34], i.e., the analysis of clickstream user data recorded during interactions with websites. Web usage mining is often process-agnostic; its main purpose is to optimize websites, for example, by adapting their content and structure to users' browsing behavior [13,19]. The primary data source for web usage mining are server UI logs that are generated in a standardized logging format like the Extended Log Format [37] and have a fixed set of attributes. These include the URL of the current and previous page request,

the resource accessed, timestamps, identifying data like the user's IP address, and technical data about the user's web browser and operating system.

UBM in Other Domains. In addition to the research areas mentioned above, interaction logs have been used as a source of data-driven insights into user behavior in several other domains, such as human-computer-interaction [14,15], information retrieval [21], and visualization [18]. The logs in these domains can take various forms, but they generally record user interactions at a much lower level of detail than the process-related UI logs that we focus on in this paper.

3   Literature Review

This paper's goal is to develop a reference data model that subsumes and integrates the commonalities of current approaches for capturing process-related UI logs, but stays flexible with regard to their differences. In this section, we review UI logs from scientific literature to identify those commonalities and differences.

3.1   Research Method

We conducted a structured literature review [23] in SpringerLink, IEEE Xplore, and ACM Digital Library. As search terms, we used "log" combined with (1) "user interact*" and "user interface" (2) "task mining" and "desktop activity mining" as common terms for high-resolution process mining, and (3) "robotic process automation" and "robotic process mining" as important applications of UI logs. We limited our search to papers written in English and published after 2015 because we focus on the current state of the art. The relevance of the initial search results was assessed based on their title and abstract. This yielded a set of potentially relevant papers, on which we performed a forward-backward-search to also cover papers that our search terms might have missed.

To ascertain the relevance of the identified papers, we scanned their full text for passages on UI logs or recording approaches for them. Papers were considered as relevant if (1) they contained a concrete UI log or (2) they described the UI log collection process in enough detail to infer the captured attributes.

Table 1. The papers found in the database search

Search Term	IEEE Xplore	ACM DL	SpringerLink
log AND "user interact*"	[24]	[12]	[2,3,7,35]
log AND "user interface"			
log AND "task mining"			[26]
log AND "desktop activity mining"			
log AND "robotic process automation"			[5,22]
log AND "robotic process mining"			

As listed in Table 1, we found 9 relevant publications in the initial search. Several papers appeared in more than one query, but are only listed under the search term that we first found them with. The forward-backward search returned another 10 publications. Although we did not explicitly search for web usage mining logs, our search returned papers about server-side and also client-side web usage mining (recording web activities by adding tracking software to a browser), but none of these met the above-listed criteria. In our review, we therefore only included one exemplary clickstream log from a process mining context: the BPI Challenge 2016 [11], in which the Dutch Employee Insurance Agency recorded eight months of user activities on their website. Our final result was hence a set of 20 relevant publications.

3.2 Results

Some of the 20 relevant publications covered the same use case and data collection approach and were therefore treated as duplicates, resulting in 12 unique approaches. The majority of papers cover RPA [2–5, 7–9, 20, 22, 24–27, 35]. Four publications [6, 10, 12, 28] focus on software process mining [33]. The remaining two are general approaches to analyzing low-level user interactions with broader applications [11, 29].

Commonalities. Although the reviewed UI logs were fairly heterogeneous, we found a set of six core attributes that are recorded in more than half of them. Table 2 indicates which of the 12 approaches include which attributes (•).

Table 2. UI log attributes as found in the literature review

Source	Action type	Target element	UI Hierarchy	Application	Input value	Timestamp
[2–5]	•	•	•	•	•	•
[6]	•	•		Single	•	•
[9]	•	•	•	•	•	•
[10]	•	•		Single		•
[11]		•	•	Single	•	•
[12]	•			Single		•
[20]	•	•	•	•		•
[8, 22]	•			•	•	•
[7, 24–27]	•	•	•	•	•	•
[28]	•	•	•	Single		•
[29]	•	•		•		•
[35]		•	•	•	•	•

1. An *action type*, which describes the action a user takes. Actions are most often divided into mouse and keyboard inputs, but some logs further distinguish between different mouse buttons, string inputs, and hotkeys. Only two logs do not record action types: the BPIC 2016 clickstream log [11] and Urabe et al. [35], who only record that an interaction has taken place.

2. The atomic *target UI element*, on which the user action is executed. This attribute is recorded in most UI logs, except for two: Dev et al. [12] only record the usage of specific functions, such as crop in a graphics editor, and Jimenez-Ramirez et al. [8,22] record click coordinates and screenshots, but only use them to match similar user actions and do not map them to target elements.
3. The *software application* that the user interacts with. This could be a web browser, an ERP system, or an office application. This attribute is always recorded when researchers track user actions across multiple applications, but is not captured when the tracking is limited to a single application.
4. One or multiple attributes that specify the location of the target element in the application's *UI hierarchy*. For example, an Excel cell is located in a worksheet (hierarchy level 1), which belongs to a workbook (hierarchy level 2) [27]. UI hierarchy attributes are included in about half the reviewed logs.
5. The *input value* that the user writes into a text field. Input values are included in about half of the reviewed logs.
6. A *timestamp*, which records the exact date and time at which the action occurred. This is recorded in all logs.

Differences. Most authors characterize user interactions through an action type, i.e., *what* the user does, and a target element, i.e., *where* they do it. However, the set of possible values for the action type, and hence the level of detail at which actions are recorded, differs considerably. For example, Agostinelli et al. [3] record aggregated action types abstracted from raw hardware input (e.g., clickButton and clickTextField), whereas Jimenez-Ramirez et al. [22] make the low-level differentiation between left, right, and middle mouse clicks. Which other attributes are included in a UI log differs between approaches: whereas timestamps and the application in focus (where applicable) are recorded in all logs, input values and information on the location of a target element within the application's UI hierarchy are included in about half of them. Examples for other, less common attributes that are only recorded in few approaches include the current value of a text field [5,9,27], user IDs [5,10,27,35], other resources involved [6,11], and associations to higher-level process steps [20,29].

Another interesting finding was that most of the reviewed UI logs are initially unlabeled, i.e., they do not have a concrete case notion [16]. In some publications, events in unlabelled logs are later grouped into cases based on different attributes. These attributes include external session IDs created automatically by a system [11] or manually by users [3,27,28], user IDs [10], or case IDs from associated higher-level event logs [20].

4   Review of Industry Solutions

To ensure broad applicability of our reference data model, we also review industry approaches for conceptualizing and capturing UI logs. Because those approaches are core to the industry solutions' functionality and business secrets,

the available material for this review may be less specific than scientific papers. Therefore, we conduct the industry review in this section as an addition to the literature review, meant to confirm and complement the established findings.

4.1 Research Method

Selection Strategy. An initial analysis indicated that RPA tools are presently the only industry solutions that collect UI logs on a large scale. Some vendors also advertise task mining capabilities, but their primary focus is on recording UI logs for the automation of routines. Because the RPA market is highly fractured and fast-moving, we could not conduct a complete review. Instead, we opted to analyze a sample of companies that can be seen as representative for the market. Therefore, we selected the companies that the 2021 Gartner Magic Quadrant RPA report[1] attributes with a "high ability to execute" and/or a "high completeness of vision": UiPath, Automation Anywhere, Microsoft Power Automate, Blue Prism, NICE, WorkFusion, Pegasystems, Appian, EdgeVerve Systems, and Servicetrace. We also included Celonis Task Mining, which is the only major product that uses UI logs primarily for low-level process mining.

Review Approach. In analyzing those eleven tools, we focused on finding the commonalities and differences between the industry logs and the scientific logs. Specifically, we wanted to know whether the industry logs capture the same set of six core attributes found in the scientific logs (commonalities) and whether the industry logs capture any other attributes that could be relevant for a widely applicable reference data model (differences). To answer those questions, we collected freely available material about the tools.[2] This included trial or demo versions, documentations, and promotional material, such as videos showcasing the recording process. After collecting the material, we had to exclude two companies from our list, Pegasystems and EdgeVerve Systems, because we could not obtain sufficient information on the functionalities of their recording software.

4.2 Results

Commonalities. For each industry solution, we analyzed whether it also records the six core attributes found in the literature review. The results are summarized in Table 3. All reviewed tools record action types, target elements, input values, applications, and timestamps. Similar to what we found in the literature review, the recordable action types differ considerably between tools.

Differences. We also examined whether the industry solutions systematically record any other attributes, but we did not find any. However, we did find a significant difference between industry logs and scientific logs in how they capture

---

[1] https://www.gartner.com/en/documents/3988021.

[2] A full list of the material that we analyzed can be found at https://gitlab.uni-mannheim.de/jpmac/ui-log-data-model/-/blob/46b363dc75b992a43398f501e3a4cb 0e755107d0/industry_review_sources.pdf.

Table 3. UI log attributes as found in the industry review

Company	Action type	Target element	UI Hierarchy	Application	Input value	Time-stamp
UiPath	•	•	Screenshots	•	•	•
MS power automate	•	•	•	•	•	•
Automation anywhere	•	•	•	•	•	•
Celonis	•	•	Screenshots	•	•	•
Blue prism	•	•	Screenshots	•	•	•
Workfusion	•	•	Screenshots	•	•	•
NICE	•	•	Screenshots	•	•	•
Appian	•	•	Screenshots	•	•	•
ServiceTrace	•	•	Screenshots	•	•	•

information on the location of elements within the UI hierarchy. In research, this information is explicitly recorded in UI log attributes, but most industry tools instead store it as screenshots outside of the log. Some tools also use the UI hierarchy to construct selectors that uniquely identify an element within an application's GUI, similar to file paths. Another difference between industry logs and scientific logs concerns the case notion. In the industry solutions, the case ID is always a task or process label that is manually added to the log. Additional business context attributes also need to be added by users and are not recorded by the tool.

5   Reference Data Model

In this section, we introduce our reference data model for user interactions. We consider a reference model to be a conceptual model that serves to be reused for the design of other conceptual models [31]. Under such a reuse-oriented conceptualization, (universal) applicability of the model is not a defining property. However, maximizing the model's application scope increases its reuse potential and therefore its value to the community. Therefore, we designed the model in an inductive or bottom-up fashion [31]: based on the commonalities and differences between existing UI logs that we found in the literature and industry reviews, we constructed a model that subsumes those commonalities, but remains flexible with regard to their differences. In the following, we first elaborate on the principles that guided our design process in Subsect. 5.1. The reference model is presented in detail in Subsect. 5.2. In Subsect. 5.3, we provide a data interchange format for UI log data as a supplement to the reference model.

5.1   Design Principles

In the literature and industry reviews, we found that the main commonality between existing UI logs are the six core attributes. The main differences between them concerned the scope, the level of abstraction, and the case notion. Based on these findings, our data model follows four fundamental design principles:

1. **Minimal set of core components**: The essential characteristics of user interactions, as found in the reviews, are modeled as the components and standard attributes of the data model. Because the model is intended to be non-specific and universally applicable, we include no other elements, thus keeping the number of components and standard attributes to a minimum.
2. **Flexible scope**: To ensure flexibility in scope, the data model can be extended with any number of additional components and all components can have an arbitrary number of attributes. Also, nearly all components and standard attributes are optional. The only non-optional component and attribute that ensure the existence of a UI log are the activity and its name.
3. **Flexible level of abstraction**: To enable user interactions to be modeled in various application contexts and at various levels of abstraction, the domain of the standard attributes in the data model, such as the action type, is left unspecified and can be determined at the point of instantiation. Furthermore, all components are modeled as classes and can be subclassed. Explicit subclasses are only defined for the target object, because they are inherent to the structure of user interfaces and the way they are embedded in ISs.
4. **No explicit case notion**: Whereas the case notion of a business processes is tied to its instances, UI logs are not inherently structured along any data dimension. The reviews have shown that they can have many possible case identifiers. The data model therefore does not include an explicit case notion. Instead, the case notion needs to be defined at the point of instantiation.

5.2 Reference Model Components

The reference data model is depicted as a UML diagram in Fig. 1. It consists of nine components, modeled as classes, and their interrelations, modeled as associations. Each class has an ID and can have any number of attributes. Some components have standard attributes that have a particular significance for user interactions. In the following, we define and explain the individual components.

Components that Define the User Interaction. In our model, user interactions have two parts. First, the *action* component with its *action type* standard attribute that describes what the user does. Common action types, as observed in the reviews, correspond to the functionalities of standard peripheral input devices, such as left or right mouse clicks, single keystrokes, or keystroke combinations for shortcuts. Higher-level distinctions are also possible. For example, when collecting data in an ERP system, actions can be divided into input actions, which make changes to a business object, and navigation actions, which only serve to navigate the GUI.

The second part of an interaction is the *target object* that the action is executed on. It is instantiated as one of four object types in the UI hierarchy, as explained below. The action type and target object together determine the central model component: the *activity*. It is uniquely defined as a combination of an action and a target object and acts as the event label, like in a traditional event log. An activity has three standard attributes: the *activity name*, an optional

input value that denotes, e.g., the string that is entered into a text field, and a *timestamp* to indicate its execution time. The activity name is determined as a function of the action type of the corresponding action and the identifier of the corresponding target object, for example, a concatenation. The timestamp is a very common attribute in traditional event logs as well as UI logs. It is, however, not a strictly required attribute in the data model, since there are alternative ways to introduce a notion of order into an event log [36].

Components that Define the UI Hierarchy. The UI hierarchy integrates the various types of UI element context data into a general structure. It consists of four components, which form a tree-shaped composition hierarchy: UI element, UI group, application, and system. The UI element and UI group levels mirror the hierarchical structure of virtually all GUIs (e.g., the document object model of a website). The application and system levels go beyond the actual GUI and position it within an IS, which makes it possible to record application- and system-level user interactions and allows the UI log to be compatible with cross-application and even cross-system UI tracking.
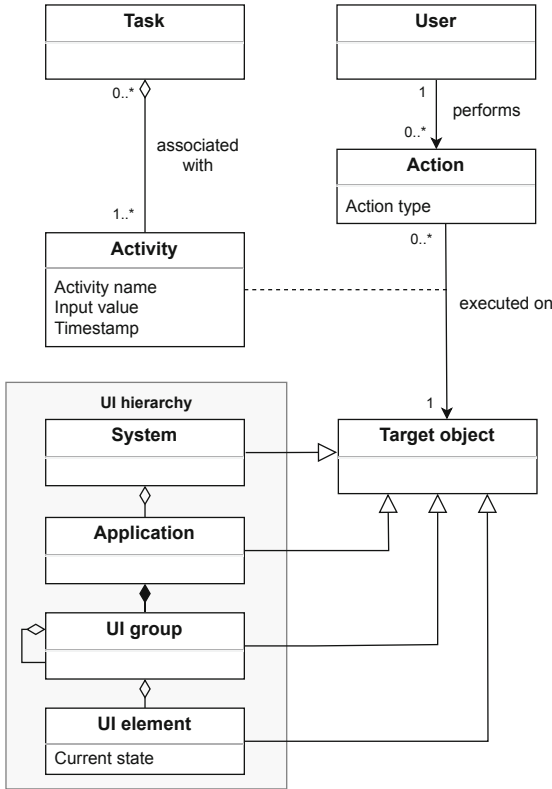
Most actions are executed on the atomic *UI elements*, which form the lowest level. Examples include buttons, text boxes, dropdowns, checkboxes, or sliders. Elements can be stateful, such as a non-empty text box or a greyed-out button. Capturing this state is necessary, for example, to track the effects of copy/paste actions or to differentiate between activity outcomes. The state of a UI element is therefore recorded in its *current state* standard attribute.

UI elements are combined into *UI groups*, which can be nested within other UI groups. In many cases, these UI groups are explicit design elements of the user interface, but our model does not impose grouping criteria and allows UI groups to be formed from arbitrary sets of UI ele-

Fig. 1. User interaction data model

ments. A simple example that we saw in the literature review is an Excel cell (UI element), which is part of a worksheet (UI group), which is again part of a workbook (UI group). Modeling UI groups has two main advantages. First, it allows to uniquely identify functionally identical UI elements. For the example above, recording information about UI groups allows us to distinguish between the cell A1 in separate Excel worksheets. This idea is used in many industry solutions to generate element selectors from screen captures. Second, UI groups can be useful for event abstraction, i.e., mapping user interactions to higher-level conceptual tasks, if these tasks are closely tied to particular UI groups. For example, all interactions with elements in a login mask (enter username, enter password, click login) can directly be abstracted to the "login" task.

UI elements and UI groups belong to an *application*, i.e., a single program instance. Some actions are directly executed on the application and are not tied to lower-level elements, such as "undo" or application-specific hotkeys.

The root node of the UI hierarchy is the *system*, on which the applications run and actions are recorded. Similar to application-level actions, it is also possible to capture system-level actions, such as the Ctrl-Alt-Del key combination to open the Task Manager on a Windows system.

Components that Define the Context. Finally, the data model includes two components that put UIs in a conceptual context: *user* and *task*. These exist in some form for all UI logs, which is why they are included in the model. In contrast, other potential context components, such as organizational or resource attributes, are use-case-specific and can be considered by extending the model.

The user is the entity that initiates any interaction. Each action is associated with a single user. Because user IDs and attributes depend on the data collection environment (e.g., device IDs in mobile applications or IP addresses on websites), the model does not specify any attributes for users. This also means that the user component is not necessarily restricted to humans and can model computer-initiated interactions, for example when recording partially automated processes.

The task component associates the recorded user interactions with conceptual tasks or routines, which makes it possible to map low-level GUI interactions to higher-level user activities. This abstraction is an essential prerequisite for being able to perform meaningful analysis on UI logs or to use them for automation.

5.3   Exchange Format

To further increase the applicability and reuse potential of our data model, we implemented it as an extension to the XES standard for event logs.[3] This *UIlog* extension provides a standardized exchange format for UI logs as a supplement to the data model. The implementation considers the activity equivalent to the event label and does not include the activity name or timestamp standard attributes because those are already provided by the concept and time extensions. The other components and standard attributes are defined at event level,

---

[3] The XML specification for the *UILog* extension is available at https://gitlab.uni-mannheim.de/jpmac/ui-log-data-model/-/raw/main/UILog_extension.

i.e., as attributes of an activity instance. The generic target object is not directly implemented, but can instead be specified through attributes that correspond to its four UI hierarchy subclasses: the target object is the lowest-level UI hierarchy component that exists for this event. For example, for an event with a UI element attribute, the target object is always this UI element, whereas for an event with no UI element or UI group attributes, the target object is the application.

6 Working Example

To demonstrate the practical utility of the reference data model, we describe in this section how it can be instantiated in a real-life scenario. The scenario is based on an RPA project, which we are currently conducting in cooperation with an ERP system vendor. The project is set in the medical technology industry, where companies are required to regularly validate their ISs to ensure that they are in compliance with external quality regulations. The validation of an IS involves manually executing a number of predefined workflows step-by-step according to a rigid execution plan, checking the result of each step against a set of acceptance criteria, and documenting the result. Manually executing a well-defined validation workflow is a repetitive and time-consuming task. The goal of our project is to automate this task using RPA. We want to record how process experts interact with the UI of the ERP system during validation and then train bots to emulate their actions.

In the following, we use the example of a keyword creation workflow to show how an artificial UI log that captures one execution of this workflow may instantiate the data model. The keyword creation workflow consists of five consecutive steps, which are executed on the GUI parts shown in Fig. 2. The user (1) logs in (a), (2) selects the right client and profile (b), (3) navigates through the dashboard (c) to reach the explorer tree (d, left), (4) creates a new keyword (d, right), and (5) logs out. The main acceptance criterion is that the newly created keyword shows up in the explorer tree after refreshing.

Table 4 shows a UI log for one case, i.e., one execution of the keyword creation workflow. It includes the action type, target UI element, and one level of UI groups, plus input value and current state where applicable. The captured action types are left and right clicks, text input, selected keyboard shortcuts, and none. The activity label of an event (in most cases) consists of the concatenated action type and target object identifier.

The first two events in the log do not correspond to single user interactions, but instead take advantage of the UI group concept to directly abstract to higher-level tasks. Instead of recording each event in the login and client selection masks separately, the task is tracked only at completion and the content of the text fields is read out when the user presses the "Login" and "Set Profile" buttons. For these abstracted activities (marked with an "A_" prefix), the action type is "none"; they are defined only through the target UI group, independent of the performed actions. This approach can be used for simple tasks with the same execution pattern in all workflows. Its main upside is that it reduces noise,

a) Login   b) Profile selection

c) Dashboard

d) Explorer tree with "Create information object" form panel

Fig. 2. The user interface of the application described in the example

which is a common problem in UI logs [26]. In our scenario, activities like initially entering a wrong password do not affect the outcome of the workflow and are therefore not relevant for automating it. By abstracting during data collection, those activities are automatically disregarded. Other advantages of abstraction are reduced implementation effort and smaller UI logs.

For effective automation, various user inputs need to be tracked. Therefore, the instantiation of the input value attribute in the log is flexible and depends on the action type and target object: When a user writes into a textbox, the input value is the entered string. When an item is selected from a dropdown, the input value records the label of that item. For abstracted activities, the input value captures the string values of all relevant UI group elements as a map.

Table 4. UI log for one execution of the keyword creation workflow

Activity	Action type	UI element	UI group	Input value	Current state
A_Login	none		login mask	{username: pren, password: dts123}	
A_Profile Selection	none		user select client	{client: base, profile: author}	
click content	left click	content	dashboard ov		
click masterdata	left click	masterdata	explorer tree		
click masterdata node expand	left click	masterdata node expand	explorer tree		
click keywords node expand	left click	keywords node expand	explorer tree		
rclick keywords	right click	keywords	explorer tree		
click ppanel new	left click	ppanel new	explorer tree		
click new information object	left click	new information object	explorer tree		
click name	left click	name	fpanel keyword		
input name	input	name	fpanel keyword	MyKeyword	
click dd type	left click	dd type	fpanel keyword		[keyword, keywords folder]
click dd type	left click	dd type	fpanel keyword	keyword	[keyword, keywords folder]
click dd linksto	left click	dd linksto	fpanel keyword		[linksto]
click dd linksto	left click	dd linksto	fpanel keyword	linksto	[linksto]
click confirm	left click	confirm	fpanel keyword		
click keywords node expand	left click	keywords node expand	explorer tree		
KEY_F5 explorer tree	KEY_F5		explorer tree		
click logout	left click	logout	explorer tree		
click confirm	left click	confirm	dialog logout		

Most state information, however, is not required for automation. Therefore, the current state attribute only records the values that can be selected from list and dropdown elements, which is needed for some more complex workflows in the validation process. For example, if a documents needs to be approved, the validation must verify that a document's author cannot be selected as approver.

This simple example demonstrates how some of the core components of the reference model can be instantiated in a real-life scenario, and how the flexibility in abstraction level can be leveraged to record attributes in a way that matches the requirements of a particular use case. It also shows that, in practice, components that are not relevant for a use case can simply be left out. The main advantage of using the reference model here is that, unlike with an ad-hoc model tailored to the use case, the attributes captured in the UI log follow a general convention that also applies to other user interfaces. This makes recording UI logs in the same format straightforward even in other applications, and makes it possible to develop automation or task mining solutions that are independent of the recording approach used.

7   Discussion and Conclusion

In this paper, we propose a reference data model for UI logs. Based on reviews of scientific literature and industry solutions, it has a set of core components to capture essential characteristics of user interactions and is flexible with regard to scope, abstraction level, and case notion. We implement the model as an XES extension and exemplarily show how it can be instantiated in a real-life RPA scenario.

Contribution. Our main objective is to address the issues that arise from the lack of standardization of UI logs. Therefore, we derive the reference from existing UI logs. Most of the components are directly adopted from the core attributes

identified in our reviews. Our contribution is their integration into a unified framework with well-defined relations. For example, we propose a rigid interpretation of an activity by defining it as a combination of an action and a target element. We also expand on the location context of UI elements and explicitly define four distinct types of target objects in an unambiguous hierarchy.

To this unified framework, we add additional, less frequently collected components and standard attributes that are particularly relevant for a complete model of user interactions. For instance, the current state is an important property of stateful UI elements when the log is intended to be used for automation. In the UI hierarchy, we add the system on top of the commonly recorded application to model system-level user interactions. We also introduce the user and task context components to add (optional) generic business context to UI logs.

Limitations. One limitation of our work concerns its grounding in existing UI logs. Despite following a methodical approach, we do not claim that our reviews or the model are complete or exhaustive. There could be unidentified UI logs or future UI logs in different use cases, which are not well represented by the model. For instance, our data model is only intended to model user interactions with graphical user interfaces, and we did not consider alternative input types, for example from voice commands or eye-tracking devices. The model may also be somewhat biased towards automation use cases because RPA solutions are overrepresented in the two reviews that it is based on.

Another limitation is that the XES standard is not particularly well-suited for UI logs. It does not support explicitly defining the relations between attributes, so all components of the UI hierarchy have to be implemented at event level. Therefore, even if many events involve the same target object, UI group, application, system and their attributes need to be included each time, leading to considerable redundancy. XES also assumes a single case notion, contrary to the flexible case notion that we intend for the data model.

Conceptually, implementing the model as an extension to the Object-Centric Event Log format OCEL [17] would be more appealing, because users, tasks, and UI hierarchy elements could be modeled as objects, reducing the redundancy. However, OCEL does not support extensions and has two main limitations with regard to UI logs. First, it does not support dynamic object attributes that can differ between events, such as the current value of a textbox that may change between interactions. Second, object attributes cannot be tied to certain object types, so for example the current state attribute cannot be limited to UI elements only. Therefore, we decided to implement the model as an XES extension.

Future Work. Our reference model can contribute to the field by providing a common, application-independent conceptual framework for user interactions. However, like any reference model, it needs to prove its utility in practice. We therefore want to encourage researchers and practitioners to adopt the model for capturing UI logs in their projects, and to extend it both with regard to new use cases and with regard to conceptual aspects, such as user privacy.

References

1. Abb, L., Bormann, C., van der Aa, H., Rehse, J.R.: Trace clustering for user behavior mining. In: European Conference on Information Systems, AIS (2022)
2. Agostinelli, S., Leotta, F., Marrella, A.: Interactive segmentation of user interface logs. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. (eds.) ICSOC 2021. LNCS, vol. 13121, pp. 65–80. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_5
3. Agostinelli, S., Leotta, F., Marrella, A.: Interactive segmentation of user interface logs. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. (eds.) ICSOC 2021. LNCS, vol. 13121, pp. 65–80. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_5
4. Agostinelli, S., Marrella, A., Mecella, M.: Automated segmentation of user interface logs. In: Robotic Process Automation, pp. 201–222. De Gruyter Oldenbourg (2021)
5. Agostinelli, S., Marrella, A., Mecella, M.: Exploring the challenge of automated segmentation in robotic process automation. In: Cherfi, S., Perini, A., Nurcan, S. (eds.) RCIS 2021. LNBIP, vol. 415, pp. 38–54. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75018-3_3
6. Ardimento, P., Bernardi, M.L., Cimitile, M., Ruvo, G.D.: Learning analytics to improve coding abilities: a fuzzy-based process mining approach. In: International Conference on Fuzzy Systems, pp. 1–7. IEEE (2019)
7. Bosco, A., Augusto, A., Dumas, M., La Rosa, M., Fortino, G.: Discovering automatable routines from user interaction logs. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNBIP, vol. 360, pp. 144–162. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26643-1_9
8. Chacón Montero, J., Jimenez-Ramirez, A., Gonzalez Enríquez, J.: Towards a method for automated testing in robotic process automation projects. In: International Workshop on Automation of Software Test, pp. 42–47 (2019)
9. Choi, D., R'bigui, H., Cho, C.: Candidate digital tasks selection methodology for automation with robotic process automation. Sustainability **13**(16), 8980 (2021)
10. Damevski, K., Shepherd, D.C., Schneider, J., Pollock, L.: Mining sequences of developer interactions in visual studio for usage smells. IEEE Trans. Softw. Eng. **43**(4), 359–371 (2017)
11. Dees, M., van Dongen, B.: BPI challenge 2016 (2016). https://data.4tu.nl/articles/dataset/BPI_Challenge_2016_Clicks_Logged_In/12674816/1
12. Dev, H., Liu, Z.: Identifying frequent user tasks from application logs. In: International Conference on Intelligent User Interfaces, pp. 263–273. ACM (2017)
13. Ding, A., Li, S., Chatterjee, P.: Learning user real-time intent for optimal dynamic web page transformation. Inf. Syst. Res. **26**(2), 339–359 (2015)
14. 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
15. Fern, X., Komireddy, C., Grigoreanu, V., Burnett, M.: Mining problem-solving strategies from HCI data. ACM Trans. Comput. Human Interact. **17**(1), 1–7 (2010)
16. Ferreira, D.R., Gillblad, D.: Discovering process models from unlabelled event logs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 143–158. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_11

17. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: a standard for object-centric event logs. In: Bellatreche, L., et al. (eds.) ADBIS 2021. CCIS, vol. 1450, pp. 169–175. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85082-1_16

18. Guo, H., Gomez, S., Ziemkiewicz, C., Laidlaw, D.: A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights. IEEE Trans. Visual. Comput. Graph. **22**(1), 51–60 (2016)

19. Ho, S., Bodoff, D., Tam, K.: Timing of adaptive web personalization and its effects on online consumer behavior. Inf. Syst. Res. **22**(3), 660–679 (2010)

20. Hofmann, A., Prätori, T., Seubert, F., Wanner, J., Fischer, M., Winkelmann, A.: Process selection for RPA projects: a holistic approach. In: Robotic Process Automation, pp. 77–90. De Gruyter Oldenbourg (2021)

21. Islamaj Dogan, R., Murray, G., Névéol, A., Lu, Z.: Understanding PubMed® user search behavior through log analysis. Database 2009 (2009)

22. 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: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 446–461. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_28

23. Kitchenham, B.: Procedures for performing systematic reviews. Keele University, Technical report (2004)

24. 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: International Conference on Process Mining, pp. 153–160. IEEE (2020)

25. Leno, V., Augusto, A., Dumas, M., La Rosa, M., Maggi, F.M., Polyvyanyy, A.: Discovering data transfer routines from user interaction logs. Inf. Syst. **107**, 101916 (2021)

26. Leno, V., Polyvyanyy, A., Dumas, M., La Rosa, M., Maggi, F.M.: Robotic process mining: vision and challenges. Bus. Inf. Syst. Eng. **63**(3), 301–314 (2021)

27. Leno, V., Polyvyanyy, A., La Rosa, M., Dumas, M., Maggi, F.: Action logger: Enabling process mining for robotic process automation. In: BPM Demos. Springer (2019)

28. Linares-Vásquez, M., White, M., Bernal-Cárdenas, C., Moran, K., Poshyvanyk, D.: Mining android app usages for generating actionable GUI-based execution scenarios. In: Working Conference on Mining Software Repositories, pp. 111–122. IEEE (2015)

29. Linn, C., Zimmermann, P., Werth, D.: Desktop activity mining - a new level of detail in mining business processes. In: INFORMATIK, pp. 245–258. Köllen (2018)

30. López-Carnicer, J.M., del Valle, C., Enríquez, J.G.: Towards an OpenSource logger for the analysis of RPA projects. In: Asatiani, A., et al. (eds.) BPM 2020. LNBIP, vol. 393, pp. 176–184. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58779-6_12

31. Rehse, J.R., Fettke, P.: A procedure model for situational reference model mining. Enterprise Model. Inf. Syst. Architect. **14**(3), 1–9 (2019)

32. Reinkemeyer, L.: Process Mining in Action: Principles. Use Cases and Outlook. Springer, Switzerland (2020). https://doi.org/10.1007/978-3-030-40172-6

33. Rubin, V.A., Mitsyuk, A.A., Lomazova, I.A., van der Aalst, W.: Process mining can be applied to software too! In: International Symposium on Empirical Software Engineering and Measurement. ACM (2014)

34. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web usage mining: Discovery and applications of usage patterns from web data. In: SIGKDD Explorations, vol. 1, pp. 12–23. ACM (2000)

35. Urabe, Y., Yagi, S., Tsuchikawa, K., Oishi, H.: Task clustering method using user interaction logs to plan RPA introduction. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) BPM 2021. LNCS, vol. 12875, pp. 273–288. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85469-0_18
36. van der Aalst, W.M.P., Santos, L.: May I take your order? In: Marrella, A., Weber, B. (eds.) BPM 2021. LNBIP, vol. 436, pp. 99–110. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94343-1_8
37. WWW Consortium: Extended log file format (1995). https://www.w3.org/TR/WD-logfile.html
38. XES Working Group: IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849 (2016)