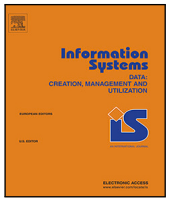
Information Systems 124 (2024) 102386

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Process-related user interaction logs: State of the art, reference model, and object-centric implementation

Luka Abb *, Jana-Rebecca Rehse

University of Mannheim, Mannheim, Germany

ARTICLE INFO

Keywords:
UI log
Data model
Robotic process automation
Task mining
Object-centric process mining

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 such a log represents an interaction between the user and the interface, such as clicking a button, ticking a checkbox, or typing into a text field. UI logs are used in many different application contexts for purposes such as usability analysis, task mining, or robotic process automation (RPA). However, UI logs suffer from a lack of standardization. Each research study and processing tool relies on a different conceptualization and implementation of the elements and attributes of user interactions. This exacerbates or even prohibits the integration of UI logs from different sources or the combination of UI data collection tools with downstream analytics or automation solutions. In this paper, our objective is to address this issue and facilitate the exchange and analysis of UI logs in research and practice. Therefore, we first review process-related UI logs in scientific publications and industry tools to determine commonalities and differences between them. Based on our findings, we propose a universally applicable reference data model for process-related UI logs, which includes all core attributes but remains flexible regarding the scope, level of abstraction, and case notion. Finally, we provide exemplary implementations of the reference model in XES and OCED.

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 [1]. Each event in a such UI log represents a single interaction between the user and the graphical user interface (GUI) of a software application. Examples for such events include clicking a button, typing into a text field, swiping left on a touchscreen, ticking a checkbox, or selecting an item from a drop-down [2]. UI logs provide a data-driven, non-intrusive, and long-term approach to studying the behavior of software users [3]. Therefore, they are used in multiple recent research streams, for example to analyze usage patterns in software applications [4–6], to identify candidate routines for robotic process automation (RPA) [2,7,8], or to derive RPA automation and test scripts [9,10]. Furthermore, software vendors such as Celonis and UiPath provide solutions designed to capture and analyze UI data, facilitating the examination and automation of specific task executions within a process [11].

When dealing with multiple UI logs from different sources, researchers and practitioners will eventually find that these logs differ substantially from one another. This inherent lack of standardization is, among other things, caused by the diverse applications of UI logs, as sketched above. Data collected for a specific research study tends to be narrowly scoped and customized to fit the proposed analysis technique or automation approach. This leads to 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. This lack of standardization also occurs in industry. Each vendor has developed their own UI log structure tailored to the capabilities of their recording software [12,13]. In addition, some vendors do not rely on commonly used log formats like CSV or XES, but generate UI logs in proprietary data formats.

The lack of standardization in UI log structure and data format causes multiple problems. First, it complicates the integration of UI logs from different sources [2,14], which might be required for purposes like cross-application automation or usability analysis. Second, it poses a challenge for the interoperability of data collection and downstream processing tools: Logs recorded by a specific data collection tool are typically only compatible with the corresponding analytics or automation methods. Combining data collection and processing tools

* Corresponding author.
E-mail address: luka.abb@uni-mannheim.de (L. Abb).

https://doi.org/10.1016/j.is.2024.102386
Received 11 January 2023; Received in revised form 15 November 2023; Accepted 7 April 2024
Available online 13 April 2024

requires considerable preprocessing effort or is entirely infeasible if the collection tool cannot record the required attributes [12]. From a research perspective, the lack of interoperability also makes it difficult to benchmark analysis and automation approaches on different datasets.

In this paper, our objective is to address the lack of standardization of UI logs, in order to facilitate the exchange and analysis of UI logs in research and practice. Therefore, we first review process-related UI logs in scientific publications and industry tools to identify and analyze commonalities and differences between them. Based on the results of this analysis, we propose a universally applicable reference data model for process-related UI logs. This reference model provides a data structure that others can reuse to conceptualize and capture UI logs in a process context. It is designed such that it subsumes and integrates the commonalities of existing logs but remains flexible with regard to the their differences regarding the logs' scope, level of abstraction, and case notion. This ensures its widespread applicability. In addition, we provide two implementations of the data model: First, we implement it as an extension for eXtensible Event Stream (XES), the current event log standard that is widely used in research. Second, we provide an exemplary UI log in two reference implementations of the Object-Centric Event Data (OCED) standard [15]. This new paradigm for the conceptualization of event logs allows to relate events with objects of different types and does not require events to be assigned to a single case [16]. As such, it fits well into the generic and flexible nature of our reference model, offering a new option for the interchange of UI logs.

We follow a reuse-oriented conceptualization of reference models, where the intended or factual reuse of a model is the only property that designates it as a reference model [17,18]. Hence, our contribution is three-fold:

1. The review of existing UI logs from research and practice identifies core components and design options of UI logs, which can be referred to when capturing and analyzing those logs.
2. The reference data model provides a blueprint for the conceptualization of UI logs, which can be instantiated in different application scenarios.
3. The exemplary implementations of the reference model in OCED demonstrate how the object-centric conceptualization of event logs can be applied to UI logs, which can help in defining a standardized interchange format for those logs in the future.

This article is an extended and revised version of our original conference publication [19]. In terms of novel content, our main extensions are the conceptualization and implementation of the reference model as object-centric event data and a critical discussion of the model's quality. In addition, we expanded the background section, repeated and updated the literature and industry reviews to ensure that our reference model captures the current state of the art, and expanded their documentation.

The remainder of this paper is structured as follows. In Section 2, we provide the necessary background information on UI logs and event logs and report on related work. Our state-of-the-art analysis is separated into two parts: First, we review existing UI logs from research in Section 3. Second, we review existing industry solutions in Section 4. Based on the commonalities and differences that we found in these reviews, we design the reference data model in Section 5. We first derive design principles, then present the model itself and its individual components. Finally, we critically discuss our design choices and their implications. In Section 6, we demonstrate how the data model can be instantiated in practice by applying it in a real-life RPA scenario, in which we record user interactions in a browser-based ERP system in order to automate simple workflows. In Section 7, we first present an implementation of the reference model as an XES extension and discuss its limitations. Then, we elaborate on the applicability of the object-centric paradigm to UI logs in general and to our data model in particular, and present our new object-centric implementations. Finally, we conclude the paper with a discussion in Section 8.

2. Background and related work

In this section, we provide the necessary background information and report on related work on event logs and exchange formats (Section 2.1), UI logs and their analysis (Section 2.2), as well as the use of UI logs in other domains (Section 2.3).

2.1. Event logs and exchange formats

The goal of process mining is to extract information about business processes through the analysis of *event logs*, i.e., collections of *events* recorded in an information system [20]. An event denotes the execution of a particular *activity* in a process, which is represented by the label of the activity (e.g., "create invoice"). In addition, events can have other attributes, such as the executing resource. In a conventional event log, these events are grouped into different *cases*, each corresponding to one process instance. A case then refers to a trace, i.e., a sequence of events that occurred during the execution of the process instance. Cases can also include additional attributes, such as the size of an order in an order-to-cash process.

To facilitate the exchange of event data between different information systems, the business process management (BPM) community has developed standardized interchange formats that define the structure, contents, and semantics of event logs. The current main format is XES, which was introduced in 2010 to replace the older MXML format [21]. In 2016, XES was accepted as the official IEEE standard for event data [22]. An updated version was released in 2023 [23].

In the XES standard, 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 XES is an IEEE standard, which is widely used in research and is supported by many process mining tools, it has major limitations. One of them is the assumption that there always exists a single case identifier that can be used to group events into clearly separated process instances. Events captured in real-life information systems often relate to multiple entities, such as orders, items, or resources [16]. Thus, there are multiple potential case identifiers and accordingly multiple perspectives that one can take on a single collection of events. To store this process data in a XES log, it needs to be flattened by selecting one of the case identifier candidates. This flattening can lead to quality issues like divergence and convergence in the log [16,24], which, for example, distort the results of automated process discovery techniques [25].

To overcome these limitations in the XES standard, the community is currently focusing on object centricity as a new paradigm for conceptualizing and capturing event data [26]. Object-centric event logs do not require a single case notion. Instead, they explicitly capture the entities (or *objects*) that an event relates to [16]. To eventually provide an object-centric alternative to XES, a working group within the IEEE Taskforce on Process Mining is currently developing conceptual foundations for Object-Centric Event Data (OCED) [26]. A preliminary OCED meta-model, shown in Fig. 1, was circulated in the community and presented at the International Conference on Process Mining's XES symposium in November 2022 [15]. Initial reference implementations [27,28] were presented the following year at the same conference. There is not yet a consensus about how exactly to implement the meta-model and which of its parts should be included in an eventual OCED standard. As a result, the reference implementations share core concepts, but differ slightly in terms of their operationalization.

In the OCED format, an event log is a collection of *events* and *objects*, which can either be physical objects, like a product, or more abstract entities, like a task or department. Both events and objects are labeled

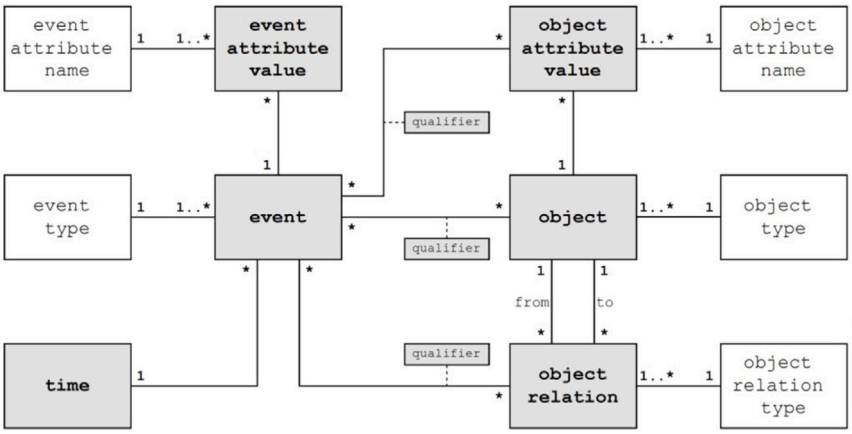


Fig. 1. The full OCED meta-model, as introduced in [15].

with a *type* and can have additional attributes. *Object relations* define how objects are related to each other. The interactions between events and objects, and their attributes and relations can be further described through *qualifiers*.

2.2. UI logs and user behavior mining

UI logs are a particular type of event log, in which events represent low-level interactions between a user of a software system and the system's GUI. For a given software system, there are two general approaches for recording UI logs. *External* logging relies on dedicated logging tools, which combine screen capture and OCR technology with a hardware input recorder [9]. It is common in industry tools. *Internal* logging draws on capabilities within the software applications, which allows recording any available internal information, but requires access the applications' source code [29]. Hence, it is more prevalent in research tools. As a hybrid between external and internal logging, application-specific plug-ins present another alternative. These plug-ins only have limited access to internal information, but can be used without source code access.

UI logs capture high-resolution data on interactions between users and software systems, which can be analyzed to generate insights into how users behave while they are engaged with an application [1,30]. The goal of such an analysis is to gain knowledge about and eventually improve the interactions between humans and IT systems. We refer to this analysis of UI logs as *user behavior mining (UBM)*. UBM can serve different purposes, such as identifying different software usage patterns [31] or increasing employee efficiency through improving the usability of a UI [32]. As such, it is closely related to several current research topics in the BPM domain.

By relating lower-level UI logs with higher-level event logs, UBM can enhance traditional process mining by providing insights into how individual process steps are executed. Process-related event data, captured for example from ERP systems like SAP or Oracle, typically records what was done in a process, such as the creation of an order. However, they do not specify how employees actually conducted these tasks. This information can be provided by recording and analyzing a corresponding UI log. This analysis of low-level process task executions is referred to as *task mining* [20] or *desktop activity mining* [33]. It can give organizations more specific insights into their processes than traditional process mining alone. It can also help software vendors to improve their products, for example, by identifying common usability issues.

Task-specific user interactions captured in a UI logs can serve as the basis for automating these tasks and the processes they belong to. For this purpose, software robots are constructed to mimic the recorded human behavior when interacting with the UI. This approach to automation is called *robotic process automation (RPA)* [29] and has lately received considerable attention in research and practice. Within RPA, UI logs are used for *robotic process mining* [2], which for example encompasses the identification of suitable tasks for automation or the segmentation of unstructured UI logs [34]. UI logs can also be used directly as an input to automation scripts, which obviates the necessity to manually configure a bot's routine.

2.3. UI logs in other domains

In addition to the research areas mentioned above, logs of interactions between users and software systems have been used as a source of data-driven insights into user behavior in several other domains. A rather prominent one is *web usage mining* [35], i.e., the analysis of clickstream user data recorded during interactions with websites. Web usage mining is often process-agnostic, meaning that it is not aware of the logic of a potential underlying (business) process. Instead, its main purpose is to improve the usability of websites, for example, by adapting their content and structure to users' browsing behavior [36,37]. The primary data source for web usage mining are server logs that are generated in a standardized logging format like the Extended Log Format [38] and have a fixed set of attributes. These attributes 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. The data recorded in these logs is limited to information transmitted from the client to the server, although it can afterwards be complemented with data from a secondary source, e.g., demographic information from a user database.

Other fields that rely on UI logs for investigating user behavior are human–computer-interaction [39–41], information retrieval [42,43], software usability [44,45], and visualization [46,47]. 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 are the focus of this paper.

3. Literature review

The first goal of this paper is to review the state of the art of process-related UI logs. The findings of this review will serve as the basis for designing the reference data model, which should subsume and integrate the commonalities between existing logs, but stay flexible with regard to their differences. In this section, we review UI logs from scientific literature to identify these commonalities and differences.

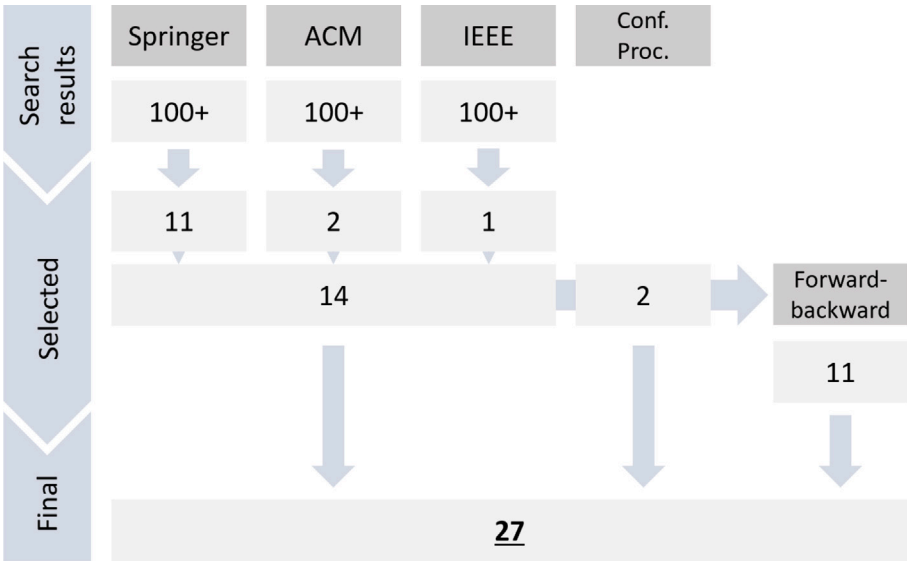


Fig. 2. The number of publications found in different stages of the literature review.

3.1. Research method

For identifying UI logs from scientific literature, we conducted a structured literature review [48] in three stages: database search, targeted search in conference proceedings, and forward–backward search. An overview of the number of publications found in each stage is shown in Fig. 2.

Database search. As we expected the relevant UI logs to mainly occur in technical papers, we selected Springer Link, ACM Digital Library, and IEEE Xplore as databases. As search terms, we used the word "log", combined with multiple other terms:

(1) "user interact*" and "user interface", as typical terms used to denote UI logs,
(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 application fields of UI logs in a process context.

Because we want to focus on the current state of the art, we limited our search to papers that were written in English and published after 2015. To ensure a certain quality, we considered only peer-reviewed journal articles and conference papers. Other than these filter criteria, we applied the default search parameters of the respective database.

This keyword search returned a set of initial results. The relevance of the papers in this set was assessed based on their title and abstract. Note that some search terms returned an excessive number of initial results; in this case, we only looked at the first 100 papers, sorting by relevance in the search interface of the respective database. This initial relevance assessment yielded a second set of potentially relevant papers. To ascertain the relevance of those papers, we scanned their full text for passages on UI logs or recording approaches for them. Papers were considered as relevant if (1) they had some relation to the Business Process Management domain and (2) they either contained a concrete UI log or described the UI log collection process in enough detail to infer the captured attributes. We excluded papers that utilize UI logs from industry tools, because these are covered in the industry review in Section 4. In the end, 14 publications were found to be relevant. The results of this initial review are summarized in Table 1. A list of all papers that we deemed potentially relevant is included in our Gitlab repository.[1]

[1] https://gitlab.uni-mannheim.de/jpmac/ui-log-data-model/-/blob/main/journal_version/Literature_Review.xlsx

Table 1
Database search summary.

	Springer Link	ACM DL	IEEE Xplore
Initial Results			
log AND "user interact*"	5685	50	32
log AND "user interface"	23 539	5184	117
log AND "task mining"	35	13	3
log AND "desktop activity mining"	6	5	0
log AND "robotic process automation"	218	43	7
log AND "robotic process mining"	18	2	2
Potentially Relevant			
log AND "user interact*"	7	0	1
log AND "user interface"	3	0	0
log AND "task mining"	7	1	0
log AND "desktop activity mining"	3	2	0
log AND "robotic process automation"	21	1	5
log AND "robotic process mining"	6	0	2
Final Selection	[2,7–9,11,29,49–53]	[10,54]	[34]

Proceedings search. Next, to supplement the database search, we searched for relevant papers directly in the proceedings of the two main process science conferences: the International Conference on Business Process Management (BPM) and the International Conference on Process Mining (ICPM). Both searches included the proceedings of workshops and associated forums, in addition to the conference proceedings themselves. In this search, we also applied the previously described exclusion and relevance criteria. This yielded another two results, one for BPM [55] and ICPM [56] respectively.

Forward–backward search. Finally, we performed a forward–backward-search on the 16 previously found papers to also cover publications that our search terms might have missed or that were published in other venues. Therefore, we checked the reference section of each paper (backward) and their respective citations using Google Scholar (forward). If any of those papers appeared relevant based on title and abstract, it was deemed potentially relevant and the full text was analyzed according to the relevance criteria listed above. This forward–backward search returned another 10 relevant 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 [57], in which

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

Source	Action type	Target element	UI hierarchy	Application	Input value	Timestamp
[9,11,49,52,53,55,58]	•	•	•	•	•	•
[6]	•	•		single	•	•
[59]	•	•	•	•	•	•
[60]	•	•		single		•
[57]		•	•	single	•	•
[4]	•			single		•
[61]	•	•	•	•		•
[10,29,50,51]	•			•	•	•
[2,7,12,34,62]	•	•	•	•	•	•
[5]	•	•	•	single		•
[33,54]	•	•		•		•
[8]		•	•	•	•	•
[56]			•	•		•

the Dutch Employee Insurance Agency recorded eight months of user activities on their website.

Our final result was hence a set of 27 relevant publications. Compared to the original conference paper, this review yielded seven additional publications [50–56].

3.2. Results

The 27 relevant publications were then analyzed to identify commonalities and differences between them. A considerable number of publications covered the same use case and data collection approach and were therefore treated as duplicates, resulting in 13 unique approaches. The majority of papers (19) cover RPA or robotic process mining [2,7–12,29,34,49–53,55,58,59,61,62]. Four publications [4–6, 60] focus on software process mining [63]. The remaining four are general approaches for the analysis of user interactions in broader applications [33,54,56,57].

3.2.1. Commonalities

Although the reviewed UI logs were fairly heterogeneous, we found a set of six core attributes that were recorded in more than half of them.

(1) An *action type* that describes the action a user takes. This is recorded in almost all logs. Actions are most often divided into mouse and keyboard inputs, but some logs further distinguish between different mouse buttons and between string inputs and hotkeys. There are three notable exceptions that do not distinguish action types: the BPIC 2016 clickstream log [57], Urabe et al. [8], and Beerepoot et al. [56].
(2) The atomic *target UI element* that the user action is executed on, for example, a button or a text field. This attribute is also recorded in all but three UI logs. One exception is the approach by Dev et al. [4], who only record the usage of specific tools (e.g., Crop) in a graphics editor application. Jimenez-Ramirez et al. [10,29] record click coordinates and screenshots, but only use them to match similar user actions and do not map them to target elements. Finally, Beerepoot et al. [56] record user interactions at a higher abstraction level than the other approaches, focusing on application windows rather than singular UI elements.
(3) One or multiple attributes that contain further information about the location of the target element in the *user interface hierarchy* of an application. An example is an Excel cell as the target element which is embedded in a worksheet (hierarchy level 1) that belongs to a workbook (hierarchy level 2) [12]. About half of the UI logs in our literature review include attributes that specify the location of the target element in the UI hierarchy.
(4) The *software application* that the user is interacting with, e.g., a web browser, ERP system, office application, etc. This attribute is always recorded when a study tracks user actions across more than one application. Some approaches, however, only record capture actions in a single application and therefore do not record it (marked as "single" in Table 2).
(5) The *input value* that the user writes into a text field. Input values are included in about half of the publications.
(6) A *timestamp* that makes it possible to establish an event order. This is recorded in all publications.

Table 2 indicates which of the 13 approaches include which attributes (•). We can conclude that the core set of attributes is fairly standardized among UI logs and should therefore be contained in a reference data model. Next, we focus on the differences between the individual logs.

3.2.2. 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. [9] record aggregated action types abstracted from raw hardware input (e.g., clickButton and clickTextField), whereas Jimenez-Ramirez et al. [29] make the low-level differentiation between left, right, and middle mouse clicks. Which other attributes are included in a UI log also differs between approaches: whereas timestamps and the application in focus (where applicable) are recorded in all logs, only about half of them include input values and information on the location of a target element within the application's UI hierarchy. Examples for other, less common attributes that are only recorded in few approaches include the current value of a text field [11,12,59], user IDs [8,11,12,60], other resources involved [6,57], and associations to higher-level process steps [33,61]. An illustration of two UI logs with different attribute sets is shown in Fig. 3.

Another interesting finding was that most of the reviewed UI logs are initially unlabeled, i.e., they do not have a concrete case notion [64]. In some publications, events in unlabeled logs are later grouped into cases based on different attributes. These attributes include external session IDs created automatically by a system [57] or manually by users [5,9,12], user IDs [60], or case IDs from associated higher-level event logs [61].

To conclude, our review of UI logs from scientific literature has found that virtually all existing UI logs rely on a set of core components that represent user interactions. At the same time, the logs differ considerably with regard to the level of detail, attribute scope, and case notion. In the next section, we study whether these findings also apply to UI logs from industry solutions.

4. Review of industry solutions

To ensure broad applicability of our reference data model, we also review the state of the art of process-related UI logs in industry to identify the commonalities and differences between them and see how they relate to the scientific logs. This additional review is presented in this section. Because those approaches are core to the industry solutions' functionality and business secrets, the available material

Row	UI Timestamp	UI Type	Payload P_1	P_2	P_3	P_4
1	2019-03-03T19:02:18	Click button (Web)	https://unimelb.edu.au	New Record	newRecord	button
2	2019-03-03T19:02:20	Select cell (Excel)	StudentRecords	Sheet1	A	2
3	2019-03-03T19:02:23	Copy cell (Excel)	StudentRecords	Sheet1	A	2
4	2019-03-03T19:02:25	Select field (Web)	https://unimelb.edu.au	Full Name	name	""
5	2019-03-03T19:02:26	Paste (Web)	https://unimelb.edu.au	Full Name	name	""
6	2019-03-03T19:02:28	Edit field (Web)	https://unimelb.edu.au	Full Name	name	text
7	2019-03-03T19:02:33	Select cell (Excel)	StudentRecords	Sheet1	B	2
8	2019-03-03T19:02:35	Copy cell (Excel)	StudentRecords	Sheet1	B	2
9	2019-03-03T19:02:42	Select field (Web)	https://unimelb.edu.au	Full Name	name	Albert
10	2019-03-03T19:02:46	Paste (Web)	https://unimelb.edu.au	Full Name	name	Albert

Timestamp	CaseId	ActivityId	MorKeyb	Coordenates	TextInput	NameApp	Screenshot
12312313	1	A	MOUSE	123,32	""	Mail client	image0001.png
12312314	1	B	MOUSE	32,43	""	Mail client	image0003.png
12312315	1	C	MOUSE	44,12	""	Mail client	image0004.png
12312316	1	D	KEYBOARD	234,367	"28362233J"	CRM	image0005.png
12312317	1	D	MOUSE	23,55	""	CRM	image0007.png
12312318	2	A	MOUSE	123,32	""	Mail client	image0008.png
12312319	2	B	MOUSE	32,43	""	Mail client	image0010.png
12312320	2	C	MOUSE	44,12	""	Mail client	image0011.png
...	...	...	...	...	...	...	...

Fig. 3. Excerpts of two UI logs from scientific publications [50,62] with different attribute sets.

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

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 2022 Gartner Magic Quadrant RPA report[2] 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, SAP Process Automation, and Salesforce MuleSoft. We have selected those tools because, based on their ranking by Gartner, they evidently have achieved a certain level of conceptual and/or technical maturity. We therefore assumed that the underlying data models were sufficiently mature, comprehensive, and stable to warrant their consideration in a state-of-the-art analysis and, eventually, their inclusion in the design of a universally applicable reference model. In addition to those tools, we also included Celonis Task Mining, which is the only major product that uses UI logs primarily for low-level process mining.

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.[3] 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 Pegasystems from our list because we could not obtain sufficient information on the functionalities of their recording software.

4.2. Results

Based on the collected materials, the remaining ten tools were analyzed to identify commonalities and differences between them and the scientific logs from Section 3. In doing so, we used the results from the previous section as a guideline.

4.2.1. 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 between tools, and in addition, separate tools often use different names for the same user action. These two aspects are illustrated in the comparison of mouse-related action types in Automation Anywhere[4] and Celonis Task Mining[5] in Table 4: both solutions support recording (single) left, right, and double clicks, but use slightly different names. Automation Anywhere does not support the recording of mouse wheel scrolling actions, whereas Celonis Task Mining does. Automation Anywhere's recorder also features an additional generic *Click* action type, which is intended to make recording more reliable through redundancy.

Additionally, the action types have considerable variation in abstraction level, even within a single tool. One tool where this phenomenon occurs is Microsoft Power Automate, as shown in Fig. 4.[6] Among the four recorded activities, the first and fourth action types are low-level hardware inputs ("click" and "populate"), enriched with information about the action's target ("element in window" and "text field in webpage"). The second and third, however, are abstracted from a series of such inputs and refer to higher-level activities, such as "launch web browser".

[2] https://www.gartner.com/en/documents/4016876

[3] A full list of the material that we analyzed can be found at https://gitlab.uni-mannheim.de/jpmac/ui-log-data-model/-/blob/main/journal_version/industry_review_sources_journal.pdf

[4] https://docs.automationanywhere.com/bundle/enterprise-v2019/page/enterprise-cloud/topics/aae-client/bot-creator/commands/cloud-recorder-object-controls-and-actions.html

[5] https://docs.celonis.com/en/schema-documentation.html

[6] Taken from https://learn.microsoft.com/en-us/power-automate/desktop-flows/automation-web.

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

Tool	Action type	Target element	UI hierarchy	Appli-cation	Input value	Time-stamp
UiPath	•	•	screenshots	•	•	•
MS Power Automate	•	•	•	•	•	•
Automation Anywhere	•	•	•	•	•	•
Celonis	•	•	screenshots	•	•	•
SS&C Blue Prism	•	•	screenshots	•	•	•
WorkFusion	•	•	screenshots	•	•	•
SAP Process Automation	•	•	•	•	•	•
NICE	•	•	screenshots	•	•	•
Appian	•	•	screenshots	•	•	•
MuleSoft	•	•	screenshots	•	•	•

Table 4
Mouse-related action types in Automation Anywhere (Universal Recorder) and Celonis Task Mining.

Automation Anywhere	Celonis Task Mining
Click	–
Left Click	Left-click
Right Click	Right-click
Double Click	Left double click
–	Mouse wheel
–	Mouse wheel (up)
–	Mouse wheel (down)

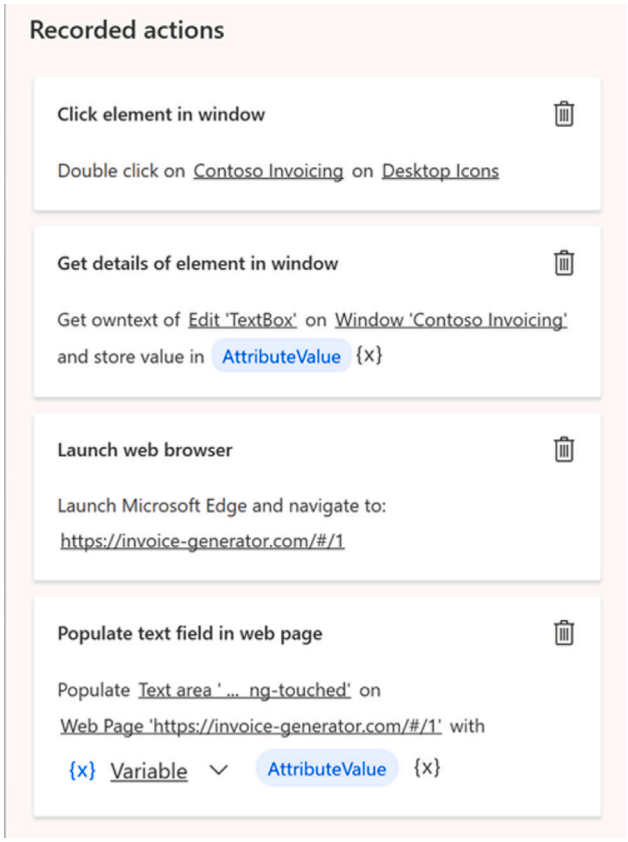


Fig. 4. Examples of recorded actions with varying abstraction levels in Microsoft Power Automate.

4.2.2. Differences

We also examined whether, in comparison to the scientific logs, the industry solutions systematically record any additional attributes, but we did not find any. However, we did find a significant difference between industry logs and scientific logs in how they capture information on the location of elements within the UI hierarchy. In research, this information is explicitly recorded in UI log attributes. Most industry tools instead store hierarchy information 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.

To conclude, our review of industry solutions has found that all existing solutions also record the set of core components that we found in the scientific logs. At the same time, there are considerable differences between the tools regarding the level of detail, set of action types, and recording of UI hierarchy information. In the next section, we build on these findings to design a reference model for process-related UI logs.

5. Reference data model

In this section, we introduce our reference data model for process-related user interaction logs. We consider a reference model to be a conceptual model that serves to be reused for the design of other conceptual models [18]. 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 [18]: 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 Section 5.1. The reference model and its individual components are presented in detail in Section 5.2. Finally, we critically discuss our design choices and their implications in Section 5.3.

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.

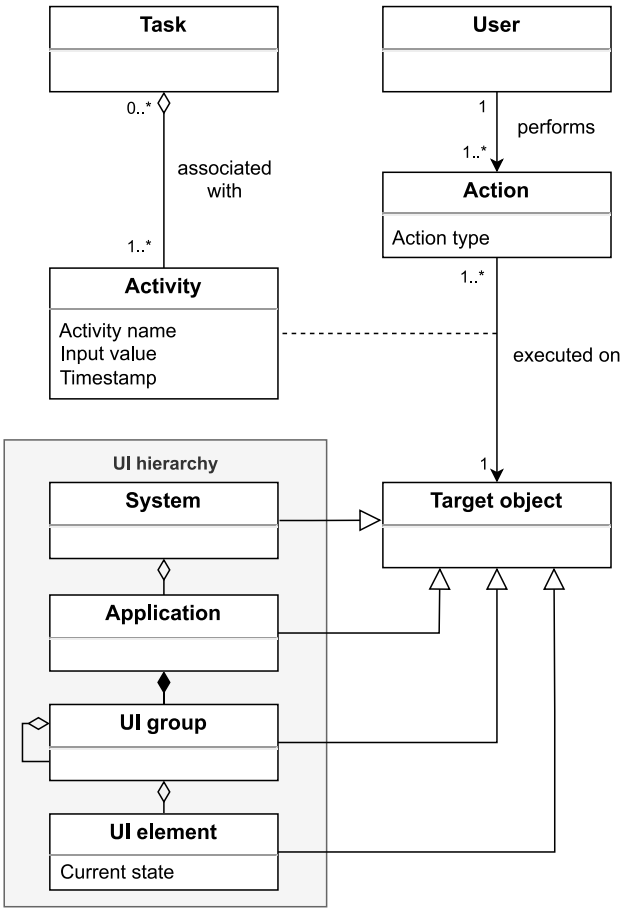


Fig. 5. User interaction log data model.

(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 information systems.

(4) **No explicit case notion**: Whereas the case notion of a business process 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

We designed the reference data model as a UML diagram, depicted in Fig. 5. 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.

5.2.1. Components that define the user interaction

In our model, user interactions consist of two parts. The first part is 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 [1]. We therefore do not specify any default values for the action type attribute.

The second part of a user interaction is the *target object* on which the action is executed. 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 user *activity*. An activity has three standard attributes: the *activity name*, which acts as the event label, like in a traditional event log, 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 can be used to calculate the duration of user interactions as well as waiting times between them. In addition, many UI logs rely on timestamps to establish an order between the individual events. However, as we wanted to keep the data model as flexible as possible and there are alternative ways to introduce a notion of order into an event log [65], we did not make timestamps a mandatory attribute of activities. In the absence of a timestamp, we would require at least one other attribute that can serve as an ordering criterion.

5.2.2. Components that define the UI hierarchy

In our reviews, we found that the target object of user interactions was logged at different levels of granularity. To incorporate them, we specified the target object in terms of a UI hierarchy, which 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 information system, 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 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 elements. 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.

5.2.3. Components that define the context

Finally, the data model includes two components that put UIs in a conceptual (process) 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 models a specific unit of work that is part of a larger process. It is used to associate 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. It is, however, also possible to record user interactions that do not belong to any task defined on a conceptual or business level; for example, an employee interacting with a social media application in between performing tasks in an ERP system.

5.3. Critical discussion of model quality

In this section, we want to briefly discuss our model's design with respect to the six quality criteria for data models proposed by Moody and Shanks [66,67]: completeness, flexibility, simplicity, understandability, integration, and implementability.

A data model is *complete* if it is able to meet all functional requirements. In our case, this means that the model should be able to capture any process-related UI log. Because our model was developed based on the set of UI logs that we found in our literature and industry reviews, it can capture all of them by design. Of course, we cannot rule out that there may be other, future UI logs that will not be completely or appropriately captured by the data model. However, at least with respect to existing logs, it can be considered as complete.

The *flexibility* of a data model denotes its ability to be adapted to differing requirements. This was one of our main priorities when developing the model and is reflected in the design principles in Section 5.1. It is important to note that this flexibility presents a trade-off: whereas adaptability is advantageous for a data model, its role as a reference model could be compromised due to the potential for varying implementations. Our model aims to facilitate the standardization of UI logs and to ensure their unification into a common format. This objective may not be fully realized if the implementations vary too widely. If, for example, two UI logs each contain a large number of additional attributes that extend the data model in different ways, or if the discrepancy in the level of abstraction among the action types becomes too significant, they would again require considerable harmonization effort before they can be compared or integrated effectively.

Simplicity of the model was another aspect that we prioritized in the design. It is achieved by including only a minimal set of components and attributes and manifests itself in a relatively low number of elements (entities and relationships) in the model. The *understandability* of a data model is closely linked to its simplicity. While we cannot conclusively evaluate this without a user study, we argue that the low complexity of the model, and its intended audience of technically proficient researchers and BPM professionals, serve as arguments in favor of its understandability.

The *integration* of a data model is achieved if it is consistent with other data, e.g., in an organizational context. For our model, it is particularly important to be consistent with other process data, e.g., higher-level events logs or process models. This is facilitated by the inclusion of the task component, which bridges the abstraction level between user interactions and process-level activities, and by the generic nature of the other components such as *User*, *UI element* or *Application*, which are independent of the environment in which the UI log is recorded and used. Beyond that, we argue that effective integration is impossible to ensure in a reference model and primarily hinges on appropriate instantiations in application-specific data models.

Finally, a good data model should be easy to implement. In analogy to the above discussion on completeness, our model's foundations in existing UI logs also supports its *implementability*, since each of these logs can be considered as an instantiation of the reference model. However, this does not inherently demonstrate its ease of implementation from the ground up. To address this, we provide a working example of an implementation of our data model in the following Section 6.

It is important to note that this discussion primarily concerns the quality of our model as a data model. Its utility as a reference model according to the reuse-oriented conceptualization of the term can be shown only through the extent of future adoption by others.

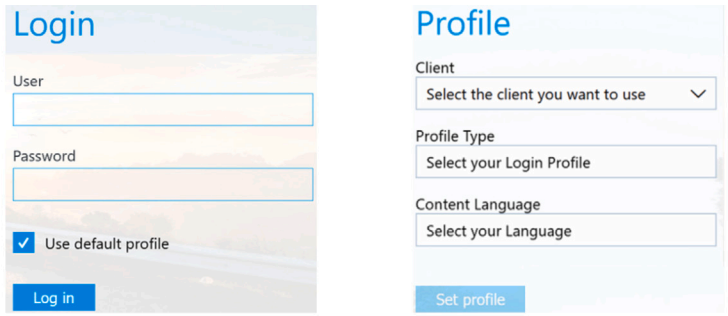
Fig. 6. Execution plan for the validation of the "Create Keyword" workflow.

6. Working example

In this section, we aim to demonstrate the practical utility of the data model by describing how we instantiated it to design the data collection process in an RPA project that we are currently conducting in cooperation with an ERP system vendor [68]. The project is set in the medical technology industry, where companies are required to regularly validate their information systems to ensure that they are in compliance with external quality regulations. The validation of an information system 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.

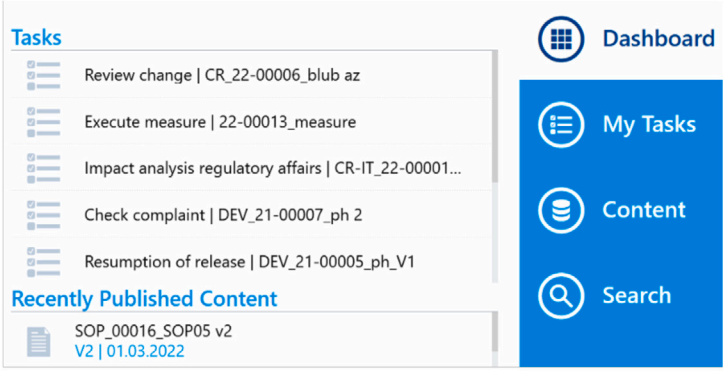
A validation project currently involves high manual effort. The goal of our case study is to automate validation for a wide range of workflows using RPA technology. We record how process experts interact with the user interface of the ERP system during validation and then train bots to emulate these interactions. To record interaction data, we modified the source code for the application's web-based front-end by adding various event handler methods to atomic and container elements (i.e., UI elements and UI groups). This allows us to capture user interactions at a high level of detail and to implement the components and standard attributes from the data model.

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. A high-level BPMN diagram of the five consecutive steps in the keyword creation workflow is shown in Fig. 6. These steps are executed on the GUI parts shown in Fig. 7.

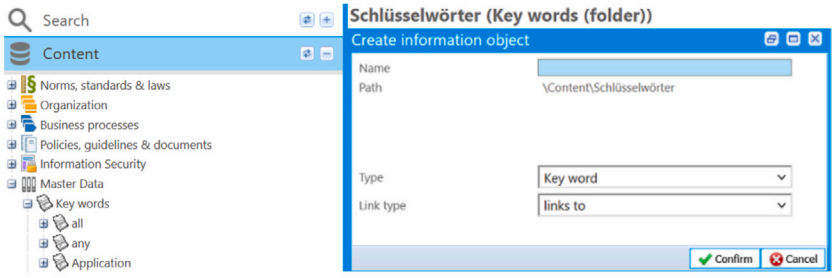


(a) Login

(b) Profile selection



(c) Dashboard



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

Fig. 7. The user interface of the application described in the example.

The user (1) logs in (Fig. 7(a)), (2) selects the right client and profile (Fig. 7(b)), (3) navigates through the dashboard (Fig. 7(c)) to reach the explorer tree (Fig. 7(d), left), (4) creates a new keyword (Fig. 7(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 5 shows an excerpt of the 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 in this log 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. 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. One upside of this approach is that it makes it possible to *not* record input values when desired. Whereas in the client selection mask, the content of the text fields is read out when the user presses the "Set Profile" button, input values are never read in the login mask to avoid recording the user's login credentials. Another advantage of this abstraction is that it reduces noise, which is a common problem in UI logs [2]. In our scenario, activities like initially entering a wrong username 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.

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. 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 approving manager.

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

Table 5
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		
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 links	left click	dd linksto	fpanel keyword		[linksto]
click dd links	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		

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.

The full log from Table 5, which contains additional attributes, is available as a CSV file in our repository. Note that we have replaced the original identifiers for UI elements, groups, and application with symbolic, human-readable ones. The main purpose of this CSV log is to demonstrate the data collected in our working example. In the next section, we provide exemplary implementations of that log in the XES and OCED formats.

7. Implementation

To increase the applicability and reuse potential of our reference data model, we also require an implementation in a common event log format. Such an implementation can be reused for the interchange of UI log data and may be the first step towards defining a standardized interchange format in event logs. Because of the current paradigm shift in the process mining community from single-case event logs towards object-centric ones, this section describes two types of implementations of our data model: a conventional, XES-based one in Section 7.1 and two object-centric, OCED-based ones in Section 7.2.

7.1. XES extension

For the first implementation of the reference model, we relied on XES as the still the de-facto interchange standard for event logs. As described above, the XES standard for event logs allows for domain-specific extensions to describe additional event log attributes. Hence, we implemented the `UILog` extension for XES, which provides a standardized exchange format for UI logs as a supplement to the data model. The XML specification for this extension is shown in Fig. 8.[7]

The implementation considers the activity equivalent to the event label. It 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, 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.

An exemplary XES version of the log described in Section 6 that implements this extension is available in our repository.

7.2. OCED implementation

As already described in Section 2, XES has multiple limitations, which play a prominent role in the implementation of UI logs. We discuss these limitations in more detail in the following subsection and explain why an object-centric implementation is well-suited for our reference data model. We then provide implementations of the UI log from Section 6 in two object-centric formats: OCEL 2.0 [27] and an OCED semantic header [28]. Of the four reference implementations of the current iteration of the OCED meta-model [15] presented at the OCED symposium at ICPM 2023, these two were the only stand-alone reference implementations. Hence, they were the only two formats that could serve our intended purpose.

7.2.1. Suitability of object-centricity for UI logs

Although it is the current event log standard, XES is not particularly well-suited for UI logs. This has two major reasons. First, XES 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, the UI group, application, system, and their respective attributes need to be included each time, causing considerable redundancy. Second, XES assumes a single case notion, contrary to the flexible case notion that we intend for the data model. To retain this flexibility, case notion candidates like the user and task context components are also implemented at event level, even though they would be case-level attributes in many UI logs.

As an alternative to XES, an object-centric exchange format could addresses these shortcomings. In an object-centric event log, all entities or objects are defined in their own right, along with their relations and attributes. An event can be related to any number of objects, eschewing redundant attribute storage. A flexible case notion is inherent to object-centric event logs, because they can be "flattened" and thus viewed from the perspective of any object type. The (full) OCED meta-model

[7] The extension is also available at https://gitlab.uni-mannheim.de/jpmac/ui-log-data-model/-/raw/main/UILog_extension.

```
<?xml version='1.1' encoding='UTF-8'?>
<xesextension name="UIlog" prefix="ui" uri="https://gitlab.
    uni-mannheim.de/jpmac/ui-log-data-model/-/raw/main/
    UILog_extension">
     <event>
         <string key="actionType">
             <alias mapping="EN" name="Action type"/>
         </string>
         <string key="uiElement">
             <alias mapping="EN" name="UI element"/>
         </string>
         <list key="currentState">
             <alias mapping="EN" name="List of attribute
                values that determine the current state of a
                stateful UI Element"/>
         </list>
         <list key="uiGroups">
             <alias mapping="EN" name="UI groups"/>
         </list>
         <string key="application">
             <alias mapping="EN" name="Application"/>
         </string>
         <string key="system">
             <alias mapping="EN" name="System"/>
         </string>
         <string key="inputValue">
             <alias mapping="EN" name="Input value"/>
         </string>
         <string key="user">
             <alias mapping="EN" name="The user initiating the
                event"/>
         </string>
         <string key="task">
             <alias mapping="EN" name="The conceptual task
                that the event belongs to"/>
         </string>
     </event>
</xesextension>
```

Fig. 8. XML specification for the *UILog* extension.

is shown in Fig. 1. The left side of the model describes events, their associated attributes, and the time construct, which are found in a similar form in XES. The right side describes objects and their attributes, as well as typed relations between objects. In the center, these two clusters are connected through qualified associations. Note that there also exists a "base" version of the meta-model, which is missing the associations between event and object relation and between event and object attribute value [69].

Our reference data model lends itself very well to an object-centric implementation, since almost all of its components can be conceptualized as objects and it already defines object-to-object relations, e.g., in the UI hierarchy, and object-to-event relations, e.g., between action and target object. Hence, the mapping of our components to the OCED meta-model is relatively straightforward. The *Activity* component in our reference model corresponds to the event in the OCED meta-model. The attributes of the *Action* and *Activity* components are event attributes. The *Activity name* and *Timestamp* standard attributes of the *Activity* component are equivalent to the event type and time concepts in OCED. The four components of the UI hierarchy (*UI element*, *UI group*, *Application*, and *System*), as well as the *User* and *Task* components, are object types. Their concrete instances in a UI log are objects. All attributes of these components are therefore object attributes. Finally, the aggregation/composition associations within the UI hierarchy are translated to object relations. For the basic version of the reference model, these are the only object relations, as the only other potential objects are *Task* and *User*, which are only indirectly connected with other objects.

If the reference model is extended with components that are specific to the domain or the use case, this may of course introduce additional object types and object relations. For example, the full version of the UI log from Section 6 features six attributes not shown in Table 5: (1) a timestamp, (2) the name of the validation test case that is being recorded (*Keyword Creation* in the working example), (3) the step within the test case and (4) its position, which together instantiate the *Task* component of the reference model (e.g., *Login* is the first step to be performed), (5) the HTML tag of the targeted UI element (where applicable), and (6) the application (*browser* in the working example). In this log, the test case and step attributes can be conceptualized as objects with a hierarchical relation (multiple steps within one test case). The position of the step within the test case is then an attribute of the step object. Likewise, the HTML tag is an attribute of the UI element object.

7.2.2. Implementation in OCEL 2.0

OCEL 2.0 [27] is a revision of the first object-centric event log standard proposed in 2021 [24]. It supports SQLite, XML and JSON

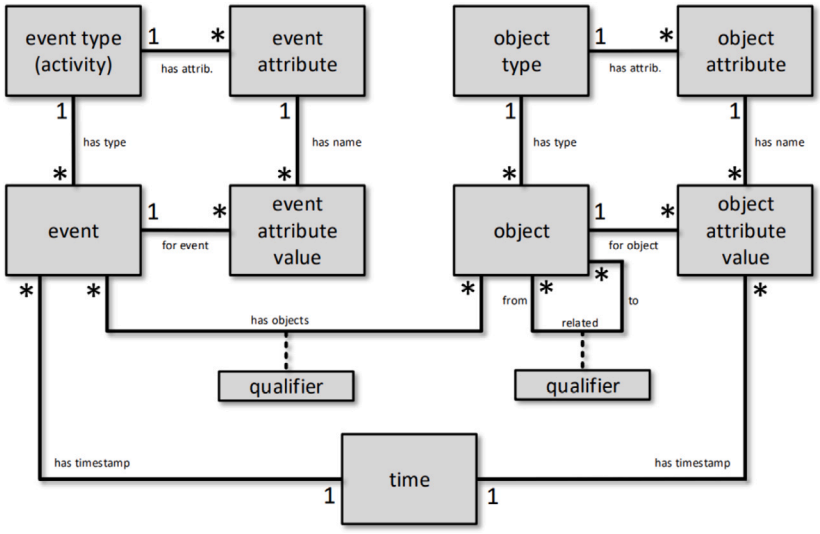


Fig. 9. The adapted version of the OCED meta-model employed in OCEL 2.0 [27].

exchange formats. OCEL 2.0 implements a slightly adapted version of the OCED meta-model, as shown in Fig. 9. In this version, event–object relationships are simplified by removing the direct associations between event and object attribute value respectively object relation. This means that the implementation does not provide a possibility to specify how an event is related to an object attribute or a relation between objects. Instead, events can only be directly related to object instances. In this sense, the implementation is closer to the above-mentioned base OCED meta-model than to the full one in Fig. 1.

In addition, object attribute values in OCEL 2.0 have a direct link to the time concept, which allows for the specification of attribute value time series independent of observed events. This makes it possible to transitively re-establish the association between events and object attribute values through the timestamp of an event, its relation to an object instance, and the timestamps of that object's attribute values.

An OCEL 2.0 JSON file that implements our working example UI log is included in our repository. This implementation maps the attributes of the working example log to event attributes, objects, or object attributes, as described in the previous subsection. Concretely, we include *activity*, *timestamp*, *action type*, and *input value* as event attributes and define five object types: *test case*, *step*, *ui element*, *ui group*, and *application*. The *step* object type has the attribute *position*; the *ui element* object type has the attributes *current state* and *html tag*. The correlation between attribute values and object instances is taken directly from the tabular source log that was produced by our recording tool, i.e., a particular object has an attribute value (at a certain time) if they are observed together in the same event. We assign each attribute value the timestamp of the event that it is observed in, because our recording tool does not include a mechanism to evaluate object attributes independent of the user actions recorded.

There are two types of object relationships in our OCEL 2.0 implementation: the hierarchy among *test case* and *step*, as well as the hierarchy among *ui element, ui group*, and *application*. In line with the OCEL schema, these are defined not at object type level, but individually for each object instance. For example, the object *masterdata* of type *ui element* has a relationship to the object *explorer tree* of type ui group, but there is no general relationship defined between *ui elements* and *ui groups*. We chose to leave the optional qualifiers for event to object or object to object relationships empty because they are purely descriptive and not part of the actual user interaction data.

Regarding the limitations of OCEL 2.0 compared to the full OCED meta-model (as described above), we argue that these are of little consequence for UI logs that follow our reference model. For the first limitation, even though there is no direct relation between events and object attribute values in OCEL either, dynamic object attributes and their connection to events are realized through the relation between object attribute values and time, so we do not consider this a limitation of the format.

For the second limitation, the missing association between event and object relation means that object relations must remain static and cannot change throughout an event log. This is in line with the relations between objects in the UI hierarchy, which are generally static: a particular UI element, such as a button, will always remain part of its UI group and will never suddenly move to a different application. This limitation could become a problem if users and tasks are both conceptualized as objects with a direct relation: in this case, it would not be possible for a user to switch from executing one task to another. However, in our reference model, there is no direct association between these two components. Rather, they are indirectly connected through the activity.

7.2.3. Semantic header implementation

The central idea of the semantic header reference implementation [28] is to keep the event log data in a format that can be directly extracted from a source system, e.g., a collection of CSV exports from a relational database. This data is then supplemented with an additional semantic header that specifies how this raw data maps to OCED concepts and how it can be transformed into an event knowledge graph [70], a data structure that naturally model events, objects, and their relations for process mining. Unlike OCEL 2.0, this reference implementation operationalizes the original OCED meta-model without modifications. In the presented implementation [28], the meta-model is translated into a property graph (PG) schema, as shown in Fig. 10. This schema can be instantiated in two JSON files: First, a dataset description file, which contains the name and data type of each attribute and specifies to which column(s) in the underlying data the attribute corresponds. Second, the actual semantic header file, which describes how the raw records are transformed into the event knowledge graph, i.e., which record attributes are used to construct event nodes, entity nodes, and their properties. It also defines relations between entities, i.e., object relations.

Although the semantic header is designed to handle collections of records (as they would be created when exporting several tables from a database), in our case, the UI recording tool only produced a single CSV file, which contains all information. To implement the semantic header, we therefore only define a single EventRecord and create the dataset description file by simply defining one record attribute for each column in the tabular UI log. We set those attributes that do not have

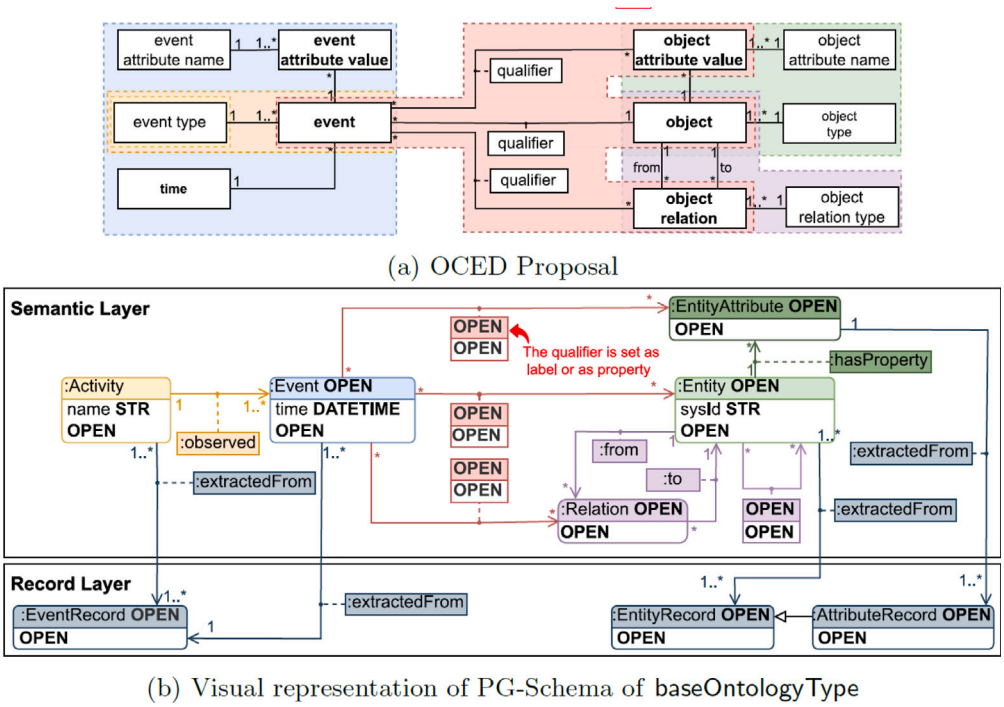


Fig. 10. Implementation of the OCED meta-model as the semantic layer of a PG schema. The event data is stored separately, visualized as the record layer. *Source:* Figure taken from [28].

a value for every event to be optional (e.g., the *input value*, which does not exist for *click* actions). In the semantic header file, we define the same four event attributes and five objects/entity nodes as in the OCEL implementation: *activity*, *timestamp*, *action type*, and *input value* as well as *test case*, *step*, *ui element*, *ui group* and *application*.

We also assign the same object attributes as in the OCEL implementation: *position* for the *step* object as well as *current state* and *html tag* for the *ui element* object. Finally, we define the same three object relations: *step* to *test case*, *ui element* to *ui group*, and *ui group* to *application*. Unlike in OCEL, these are defined at the level of object types instead of object instances. Which object instances are related to each other is inferred from their co-occurrence in events in the underlying record. We also specify in the header that all other edges in the resulting event knowledge graph (such as the directly-follows relations between events) should likewise be inferred from the raw data.

The dataset description and header JSON files for our working example UI log are also included in our repository.

8. Discussion and conclusion

In this paper, we address the lack of standardization of UI logs, in order to facilitate the exchange and analysis of UI logs in research and practice. Based on a review of the state of the art of process-related UI logs in scientific literature and industry solutions, we propose a reference data model for UI logs. This model consists of 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 exemplarily show how the model can be instantiated in a real-life RPA scenario and implement it in both a conventional and two object-centric formats. This way, we demonstrate that the emerging paradigm of object-centricity for event data is well suitable for being applied to UI logs.

To achieve our main objective, i.e., address the issues that arise from the lack of standardization of UI logs, we prioritize the future reuse of our reference model and accompanying implementations. Therefore, we derive the model from existing UI logs and show how it can be implemented in multiple standardized data interchange formats. Most of the model components (activities, action types, UI elements, UI groups, applications, timestamps, and input values) are directly adopted from the core attributes identified in our literature and industry 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, which is collected in various forms by most approaches, 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.

By providing a domain-specific event log in two reference implementations of the current OCED meta-model, we also make a contribution to the ongoing development of the OCED standard. It can serve as an example and starting point for other implementations, and our discussion of the implementations design choices and limitations in the context of UI logs will hopefully help shape the discussion about the requirements that an object-centric exchange format needs to fulfill.

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. However, our literature review also included UI logs that are not created in an automation context, for example, the ones used by Beerepoot et al. which are used to understand people's high-level work practices. Our

goal was to scope the data model in a way that it is not limited to RPA but pertains to all potential use cases of UI logs that we identified.

A second limitation is that our object-centric implementations are based on a standard that is not yet finalized and are thus only preliminary. It is possible that they will become outdated once the development of OCED progresses further.

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.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research was supported in part by the German Federal Ministry for Education and Research, grant number 01IS21044B. We would like to thank the senior PC member Andrea Marrella and the three anonymous reviewers for their helpful comments and support on our original conference paper. Furthermore, our gratitude goes to Dirk Fahland and Marco Montali for the in-depth discussion of our object-centric implementations.

References

[1] L. Abb, C. Bormann, H. van der Aa, J.-R. Rehse, Trace clustering for user behavior mining, in: European Conference on Information Systems, AIS, 2022.
[2] V. Leno, A. Polyvyanyy, M. Dumas, M. La Rosa, F.M. Maggi, Robotic process mining: Vision and challenges, Business & Information Systems Engineering 63 (3) (2021) 301–314.
[3] S. Dumais, R. Jeffries, D. Russell, D. Tang, J. Teevan, Understanding user behavior through log data and analysis, in: Ways of Knowing in HCI, Springer, 2014, pp. 349–372.
[4] H. Dev, Z. Liu, Identifying frequent user tasks from application logs, in: International Conference on Intelligent User Interfaces, ACM, 2017, pp. 263–273.
[5] M. Linares-Vásquez, M. White, C. Bernal-Cárdenas, K. Moran, D. Poshyvanyk, Mining android app usages for generating actionable GUI-based execution scenarios, in: Working Conference on Mining Software Repositories, IEEE, 2015, pp. 111–122.
[6] P. Ardimento, M.L. Bernardi, M. Cimitile, G.D. Ruvo, Learning analytics to improve coding abilities: A fuzzy-based process mining approach, in: International Conference on Fuzzy Systems, IEEE, 2019, pp. 1–7.
[7] A. Bosco, A. Augusto, M. Dumas, M. La Rosa, G. Fortino, Discovering automatable routines from user interaction logs, in: BPM Forum, Springer, 2019, pp. 144–162.
[8] Y. Urabe, S. Yagi, K. Tsuchikawa, H. Oishi, Task clustering method using user interaction logs to plan RPA introduction, in: Business Process Management, Springer, 2021, pp. 273–288.
[9] S. Agostinelli, M. Lupia, A. Marrella, M. Mecella, Automated generation of executable RPA scripts from user interface logs, in: BPM Forum, Springer, 2020, pp. 116–131.
[10] J. Chacón Montero, A. Jimenez-Ramirez, J. Gonzalez Enríquez, Towards a method for automated testing in robotic process automation projects, in: International Workshop on Automation of Software Test, 2019, pp. 42–47.
[11] S. Agostinelli, A. Marrella, M. Mecella, Exploring the challenge of automated segmentation in robotic process automation, in: Research Challenges in Information Science, Springer, 2021, pp. 38–54.
[12] V. Leno, A. Polyvyanyy, M. La Rosa, M. Dumas, F. Maggi, Action logger: Enabling process mining for robotic process automation, in: BPM Demos, Springer, 2019.
[13] M. Völker, M. Weske, Conceptualizing bots in robotic process automation, in: Conceptual Modeling, Springer, 2021, pp. 3–13.
[14] J.M. López-Carnicer, C. del Valle, J.G. Enríquez, Towards an OpenSource logger for the analysis of RPA projects, in: BPM Forum, Springer, 2020, pp. 176–184.
[15] J. Lebherz, C. Di Ciccio, OCED meta-model presentation, XES/OCED symposium, ICPM 2022, 2022, https://icpmconference.org/2022/wp-content/uploads/sites/7/2022/12/OCED_Symposium_intro.pdf.
[16] W.M.P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods, Springer, 2019, pp. 3–25.
[17] J. vom Brocke, Design principles for reference modeling: Reusing information models by means of aggregation, specialisation, instantiation, and analogy, in: P. Fettke, P. Loos (Eds.), Reference Modeling for Business Systems Analysis, Idea Group Publishing, 2007, pp. 47–75.
[18] J.-R. Rehse, P. Fettke, A procedure model for situational reference model mining, Enterprise Model. Inf. Syst. Archit. 14 (3) (2019).
[19] L. Abb, J.-R. Rehse, A reference data model for process-related user interaction logs, in: Business Process Management, Springer, 2022, pp. 57–74.
[20] L. Reinkemeyer, Process Mining in Action: Principles, Use Cases and Outlook, Springer, 2020.
[21] H. Verbeek, J. Buijs, B. van Dongen, W. van der Aalst, XES, XESame, and ProM 6, in: Advanced Information Systems Engineering, Springer, 2010, pp. 60–75.
[22] XES Working Group, IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams, 2016, IEEE Std 1849.
[23] XES Working Group, IEEE approved draft standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams, 2023, IEEE Std.
[24] A. Ghahfarokhi, G. Park, A. Berti, W. van der Aalst, OCEL: A standard for object-centric event logs, in: Advances in Databases and Information Systems, Springer, 2021, pp. 169–175.
[25] D. Fahland, Artifact-centric process mining, in: Encyclopedia of Big Data Technologies, Springer, 2019, pp. 108–117.
[26] M.T. Wynn, J. Lebherz, W. van der Aalst, R. Accorsi, C. Di Ciccio, L. Jayarathna, H. Verbeek, Rethinking the input for process mining: Insights from the XES survey and workshop, in: ICPM Workshops, Springer, 2021.
[27] A. Berti, I. Koren, J.N. Adams, G. Park, B. Knopp, N. Graves, M. Rafiei, L. Liß, L. Tacke Genannt Unterberg, Y. Zhang, C. Schwanen, M. Pegoraro, W.M. van der Aalst, OCEL (object-centric event log) 2.0 specification, 2023, URL https://www.ocel-standard.org/2.0/ocel20_specification.pdf.
[28] A. Swevels, D. Fahland, M. Montali, Implementing object-centric event data models in event knowledge graphs, in: ICPM Workshops, 2023, pp. 431–443.
[29] A. Jimenez-Ramirez, H.A. Reijers, I. Barba, C. Del Valle, A method to improve the early stages of the robotic process automation lifecycle, in: Advanced Information Systems Engineering, Springer, 2019, pp. 446–461.
[30] J.-R. Rehse, L. Abb, G. Berg, C. Bormann, T. Kampik, C. Warmuth, User behavior mining: A research agenda, Bus. Inf. Syst. Eng. (2024).
[31] S. Pachidi, M. Spruit, I. Van De Weerd, Understanding users' behavior with software operation data mining, Comput. Hum. Behav. 30 (2014) 583–594.
[32] S. Astromskis, A. Janes, M. Mairegger, A process mining approach to measure how users interact with software: An industrial case study, in: International Conference on Software and System Process, ACM, 2015, pp. 137–141.
[33] C. Linn, P. Zimmermann, D. Werth, Desktop activity mining - A new level of detail in mining business processes, in: INFORMATIK, Köllen, 2018, pp. 245–258.
[34] V. Leno, A. Augusto, M. Dumas, M. La Rosa, F.M. Maggi, A. Polyvyanyy, Identifying candidate routines for robotic process automation from unsegmented UI logs, in: International Conference on Process Mining, IEEE, 2020, pp. 153–160.
[35] J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: Discovery and applications of usage patterns from web data, in: SIGKDD Explorations, vol. 1, ACM, 2000, pp. 12–23.
[36] S. Ho, D. Bodoff, K. Tam, Timing of adaptive web personalization and its effects on online consumer behavior, Inf. Syst. Res. 22 (3) (2010) 660–679.
[37] A. Ding, S. Li, P. Chatterjee, Learning user real-time intent for optimal dynamic web page transformation, Inf. Syst. Res. 26 (2) (2015) 339–359.
[38] WWW Consortium, Extended log file format, 1995, URL https://www.w3.org/TR/WD-logfile.html.
[39] S. Dumais, R. Jeffries, D. Russell, D. Tang, J. Teevan, Understanding user behavior through log data and analysis, in: Ways of Knowing in HCI, Springer, 2014, pp. 349–372.
[40] A. Marrella, L.S. Ferro, T. Catarci, An approach to identifying what has gone wrong in a user interaction, in: Human-Computer Interaction, Springer, 2019, pp. 361–370.
[41] X. Fern, C. Komireddy, V. Grigoreanu, M. Burnett, Mining problem-solving strategies from HCI data, ACM Trans. Comput.-Hum. Interact. 17 (1) (2010).
[42] R. Islamaj Dogan, G. Murray, A. Névéol, Z. Lu, Understanding PubMed® user search behavior through log analysis, Database 2009 (2009).
[43] N. O'Hare, P. Juan, R. Schifanella, Y. He, D. Yin, Y. Chang, Leveraging user interaction signals for web image search, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2016, pp. 559–568.
[44] W. Jorritsma, F. Cnossen, R. Dierckx, M. Oudkerk, P. Van Ooijen, Pattern mining of user interaction logs for a post-deployment usability evaluation of a radiology PACS client, Int. J. Med. Inform. 85 (2015) http://dx.doi.org/10.1016/j.ijmedinf.2015.10.007.

[45] L. Torok, M. Pelegrino, J. Lessa, D. Trevisan, C. Vasconcelos, E. Clua, A. Montenegro, Evaluating and customizing user interaction in an adaptive game controller, in: International Conference of Design, User Experience, and Usability, Springer, 2015.
[46] H. Guo, S. Gomez, C. Ziemkiewicz, D. Laidlaw, A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights, IEEE Trans. Vis. Comput. Graphics 22 (1) (2016) 51–60.
[47] F. Dabek, J.J. Caban, A grammar-based approach for modeling user interactions and generating suggestions during the data exploration process, IEEE Trans. Vis. Comput. Graphics 23 (1) (2017) 41–50.
[48] B. Kitchenham, Procedures for Performing Systematic Reviews, Tech. Rep. 33, Keele University, 2004.
[49] S. Agostinelli, F. Leotta, A. Marrella, Interactive segmentation of user interface logs, in: Service-Oriented Computing, Springer, 2021, pp. 65–80.
[50] A. Martínez-Rojas, A. Jiménez-Ramírez, J.G. Enríquez, H.A. Reijers, Analyzing variable human actions for robotic process automation, in: Business Process Management, Springer, 2022, pp. 75–90.
[51] 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: Business Process Management: Blockchain, Robotic Process Automation and Educators Forum, Springer, 2023, pp. 120–134.
[52] F. Rybinski, S. Schüler, Process discovery analysis for generating RPA flowcharts, in: Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum, Springer, 2022, pp. 231–245.
[53] S. Agostinelli, M. Lupia, A. Marrella, M. Mecella, SmartRPA: A tool to reactively synthesize software robots from user interface logs, in: Intelligent Information Systems, Springer, 2021, pp. 137–145.
[54] L. Kornahrens, S. Kritzler, D. Werth, Extracting process instances from user interaction logs, in: 4th International Conference on Advanced Information Science and System, ACM, 2023.
[55] S. Agostinelli, G. Acitelli, M. Capece, M. Mecella, A human-in-the-loop approach to support the segments compliance analysis, in: Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum, Springer, 2022, pp. 200–214.
[56] I. Beerepoot, D. Barenholz, S. Beekhuis, J. Gulden, S. Lee, X. Lu, S. Overbeek, I. van de Weerd, J.M. van der Werf, H.A. Reijers, A window of opportunity: Active window tracking for mining work practices, in: International Conference on Process Mining, IEEE, 2023, pp. 57–64.
[57] M. Dees, B. van Dongen, BPI challenge 2016, 2016, URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2016_Clicks_Logged_In/12674816/1.
[58] S. Agostinelli, A. Marrella, M. Mecella, Automated segmentation of user interface logs, in: Robotic Process Automation, De Gruyter Oldenbourg, 2021, pp. 201–222.
[59] D. Choi, H. R'bigui, C. Cho, Candidate digital tasks selection methodology for automation with robotic process automation, Sustainability 13 (16) (2021) 8980.
[60] K. Damevski, D.C. Shepherd, J. Schneider, L. Pollock, Mining sequences of developer interactions in visual studio for usage smells, IEEE Trans. Softw. Eng. 43 (4) (2017) 359–371.
[61] A. Hofmann, T. Prätori, F. Seubert, J. Wanner, M. Fischer, A. Winkelmann, Process selection for RPA projects: A holistic approach, in: Robotic Process Automation, De Gruyter Oldenbourg, 2021, pp. 77–90.
[62] V. Leno, A. Augusto, M. Dumas, M. La Rosa, F.M. Maggi, A. Polyvyanyy, Discovering data transfer routines from user interaction logs, Inf. Syst. (2021).
[63] V.A. Rubin, A.A. Mitsyuk, I.A. Lomazova, W. van der Aalst, Process mining can be applied to software too!, in: International Symposium on Empirical Software Engineering and Measurement, ACM, 2014.
[64] D.R. Ferreira, D. Gillblad, Discovering process models from unlabelled event logs, in: Business Process Management, Springer, 2009.
[65] W. van der Aalst, L. Santos, May I take your order? in: BPM Workshops, Springer, 2022, pp. 99–110.
[66] D.L. Moody, G.G. Shanks, What makes a good data model? Evaluating the quality of entity relationship models, in: International Conference on the Entity Relationship Approach, Springer, 1994, pp. 94–111.
[67] D.L. Moody, G.G. Shanks, Improving the quality of data models: Empirical validation of a quality management framework, Inf. Syst. 28 (6) (2003) 619–650.
[68] N. Elsayed, L. Abb, H. Sander, J.-R. Rehse, Automating computer software validation in regulated industries with robotic process automation, in: Business Process Management: Blockchain, Robotic Process Automation and Educators Forum, Springer, 2023, pp. 135–148.
[69] TFPM OCED working group, OCED call for action: Reference implementations, 2023, URL https://www.tf-pm.org/upload/1678694478319.pdf.
[70] D. Fahland, Process mining over multiple behavioral dimensions with event knowledge graphs, in: Process Mining Handbook, Springer, 2022, pp. 274–319.