XRFlow: Visualizing Metrics and Insights for Extended Reality

Johannes Günter Herforth
University of Luxembourg
Esch-sur-Alzette, Luxembourg
johannes.herforth@uni.lu

Fiammetta Zanetti
University of Luxembourg
Esch-sur-Alzette, Luxembourg
fiammetta.zanetti@uni.lu

Karsten Schönbein
University of Luxembourg
Esch-sur-Alzette, Luxembourg
karsten.schonbein@uni.lu

Annika P.C. Lutz
University of Luxembourg
Esch-sur-Alzette, Luxembourg
annika.lutz@uni.lu

Jean Botev
University of Luxembourg
Esch-sur-Alzette, Luxembourg
jean.botev@uni.lu

Abstract

Extended Reality (XR) applications combine virtual and physical elements, generating continuous data streams that enable us to analyze interactions between the users and the virtual objects. With the extensive volume of physiological and behavioral information, processing and obtaining an overview of the recorded data is often difficult and time consuming. Existing approaches typically replay recorded sessions or deploy fixed analysis pipelines. Such approaches risk overlooking time-critical or out-of-view interactions, as pipelines are only designed to detect the specific conditions they were designed to capture. This paper introduces XRFlow, an object-centric data exploration tool comprising two components: a Unity plugin that simplifies the acquisition of object- and event-based data, and an external visualization interface that renders both these data and auxiliary data streams from objects within the environment. Using a simulated scenario, we illustrate how exploratory analysis through XRFlow can uncover subtle errors and provide comprehensive overviews of XR sessions for developers and researchers.

CCS Concepts

• **Human-centered computing** → **Visualization toolkits**; • **Information systems** → **Data streaming**; **Data analytics**; • **Applied computing** → *Psychology*; • **Computing methodologies** → *Mixed / augmented reality*.

Keywords

Extended Reality, Data visualization, 3D environments, Exploratory analyses, Object-centric process mining

ACM Reference Format:
Johannes Günter Herforth, Fiammetta Zanetti, Karsten Schönbein, Annika P.C. Lutz, and Jean Botev. 2026. XRFlow: Visualizing Metrics and Insights for Extended Reality. In *International Workshop on Immersive Mixed and Virtual Environments Systems (MMVE' 26), April 4–8, 2026, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3798090.3799677

MMVE' 26, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2535-7/26/04
https://doi.org/10.1145/3798090.3799677

1 Introduction

While data collection in Extended Reality (XR) is straightforward, the subsequent review and analysis remain a significant challenge. Existing workflows often focus on visual playback, a process that is inherently slow, as it requires manual inspection of full replays to locate regions of interest. This reliance on replays obscures subtle, non-visual errors and limits analyses around the user's field of view.

The lack of an overview tool creates a difficult environment for the data controller responsible for managing and reviewing XR session data. This role includes developers verifying that a session recorded the correct data in the proper order, or researchers checking system integrity and physiological trends throughout a user study. To ensure experimental integrity and implement quality assurance measures, controllers need interfaces that minimize the risk of human error [8]. However, maintaining this integrity is difficult when data is fragmented across multiple files and pipelines. This lack of links between files forces a manual redefinition of the workflow for each project, increasing the likelihood of processing mistakes and inconsistent results.

To address these challenges, we apply process mining techniques to XR data. By representing an XR session as a structured event log, the data can be segmented into contextually relevant traces to specific time periods. While common in business process intelligence, these techniques are rarely applied to the naturally event-heavy processes found in XR applications. We adopt an object-centric approach, enabling the data controller to isolate the behavior of specific entities, such as a user's hand or an interactive target. By focusing on objects and their events directly, this method replaces the need for custom pipelines to convert a disconnected set of files into a coherent, multi-perspective map of the session.

Current tools are not built for this iterative review workflow. Existing process mining applications often have a high entry barrier and require conceptual knowledge to set up. Additionally, while these tools are made for event-specific data, they are rarely built to handle multimodal formats, like physiological data, and often provide non-interactive environments as visualizations. Similarly, statistical tools require additional knowledge of time-series data to manually trim, prepare, and synchronize different data ranges. This fragmentation means that there is no single tool that enables the immediate exploration of both types in combination.

We present XRFlow, a system designed to bridge the gap between XR data and quick insights. It addresses the core challenges by showing the event data in a fully interactive environment, providing

different methods of viewing different levels of sub traces and perspectives from different objects. To support this, we include a reference implementation of a logging system for Unity[1]. This implementation demonstrates a lightweight way to generate the additional data required for analysis and is designed to be easily replicated in other game engines or environments.

The design of XRFlow is guided by a threefold philosophy:

- **Easy Integration:** No complex setup required to begin capturing and reviewing data.
- **Accessibility:** The interface is intuitive enough for any data controller to use without specialized training.
- **Instant Feedback:** Minimize processing time to incentivize deeper exploration and faster discovery.

To demonstrate these concepts, we first review existing tools and introduce concepts of process mining in Section 2. Section 3 presents the system architecture of XRFlow, including our Unity-based reference implementation for event data logging, using a representative XR interaction task as a running example. Section 4 then illustrates the tool's utility through practical applications focusing on debugging and data analysis. We conclude by discussing the broader potential of this approach, while also addressing current limitations and outlining future work in Sections 5 and 6.

2 Background and Related Work

Current XR analysis frameworks can be broadly categorized into playback-driven and user-centric systems. Commercial platforms such as Cognitive3D[2] and research-oriented tools [9, 12] primarily focus on spatial replays and linear timelines. These systems revolve around manually scrubbing through a session to identify patterns, essentially treating the 3D data as a high-dimensional video recording. Despite offering flexibility and expressive visuals, this approach remains largely qualitative and informal. Without underlying mathematical foundations, the controller is still limited to subjective observation rather than rigorous data mining.

Some frameworks have integrated physiological data, such as Plume [10] and VRSTK [5], although they often remain restricted by proprietary log requirements or specific hardware dependencies. By not supporting generic file types for time-series data, these systems effectively lock out researchers whose existing hardware or software configurations cannot use the built-in recorder.

The fundamental limitation across these existing solutions is their focus on visual replays over the underlying event data structure. By optimizing for the player's perspective and their chronological timeline, these tools prioritize the visual reconstruction of the environment over the underlying control flows. Consequently, this focus obscures the connections between object interactions and physiological responses that occur through events. Furthermore, the requirement for specific recording frameworks reduces portability, making it difficult for researchers to apply these analysis techniques to existing projects without significant technical refactoring.

To address these limitations, we seek a mathematical basis for XR data analysis through the application of process mining. This

[1]https://unity.com/
[2]https://cognitive3d.com/

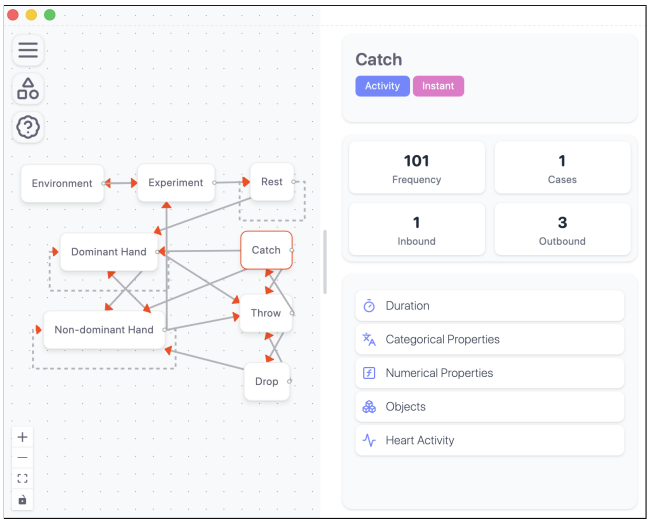


Figure 1: XRFlow user interface.

methodology is defined by three primary tasks [17]: process discovery, which generates models defining activities and their relationships; conformance checking, which compares event cases against pre-defined models; and model enhancement, where process models are supplemented with external metrics, such as physiological or temporal data, to identify hotspots of high cognitive load or process delay. Integrating these methods introduces formal methods to deal with XR sessions, using established research from other domains.

For instance, existing model similarity techniques [16] can be adapted to systematically compare interaction patterns between participants or experimental conditions. Furthermore, event abstraction methods [6, 19] can address the high granularity of XR data by clustering low-level events into higher-level behavioral definitions.

Although process mining has already been applied in software development [15] and VR environments [1, 4, 14], these implementations typically rely on flat event logs. To address the complexity of XR, Object-Centric Process Mining (OCPM) [18] provides a framework for relating single events to multiple objects, capturing non-linear interactions inherent in 3D spaces. The Object-Centric Event Log (OCEL) standard [7] further provides a unified format for these events, establishing new analysis methods for the log and models [3]. Overall, shifting towards OCPM expands the analytical capabilities for XR research, creating a more formal methodology for behavioral analysis.

3 The XRFlow System

XRFlow fills a missing use case in XR session analysis by delivering a simple platform that visualizes data from the perspective of all objects along with their events and lifecycles. By interacting with these views, controllers can detect anomalies, map inter-object relationships, and translate the resulting insights into foundations for further analyses.

We accompany XRFlow with a Unity component that illustrates how to attach a lightweight logger to any Unity-based XR scene,

making data capture straightforward and demonstrating how loggers can be developed for other engines.

To illustrate the tool's features, we utilize a running example based on a ball-catching experiment. In this scenario, participants undergo a series of trials where they must catch virtual balls using either their dominant or non-dominant hand. The dataset includes a simulated Electrocardiogram (ECG) to demonstrate how physiological data can be compared within each condition.

This section details the key features of the tool and their implementation. First, we explore the supported data formats and the custom Unity logger. Next, we detail the three main views and how the shared statistical panel integrates across them. Finally, we use the ball-catching scenario to demonstrate how these features help identify patterns and errors in the collected data.

3.1 Data Acquisition and File Support

XRFlow processes multimodal datasets through a standardized schema that pairs discrete event logs with continuous time-series data, ensuring compatibility with any XR recording system that adheres to its object-centric structure.

3.1.1 Event Log and Unity Logger. To represent complex processes, XRFlow uses an object-centric schema based on a CSV version of object-centric event logs [7]. The main format is formalized as a five-tuple structure, *[ID, Time, Activity, Lifecycle, Objects]*, adhering to the base object-centric specification while reusing the lifecycle semantics found in XES extensions.

Table 1 provides a representative subset of the log from the running example. In the sequence, a two-minute *Rest* phase is followed by a transition into the *Dominant Hand* condition. While the condition changes include lifecycle attributes, *Throw* and *Catch* activities are recorded as instantaneous events, i.e., without duration, and repeat throughout the remainder of the condition.

Beyond timing, the log identifies the objects involved to provide context for each event. For example, the *Throw* event can be interpreted as *Machine1* as the source throwing *Ball1* at *User1*. Altogether, these relationships provide perspectives that allow for later reorganization of the data around different timing and object-centric views.

Table 1: Sample event data from the running example.

ID	Time	Activity	Lifecycle	Objects
3	01:01	Rest	start	{User1}
4	03:01	Rest	complete	{User1}
5	03:05	Dominant Hand	start	{Env, User1, Machine1}
6	03:15	Throw	-	{User1, Machine1, Ball1}
7	03:16	Catch	-	{User1, Ball1}

The Unity logger uses two components designed to balance easy integration with minimal performance overhead.

The *EventLogManager* manages the global logging lifecycle, including initialization, data aggregation, and export. At startup, the manager initializes all *EventObject* instances with IDs and establishes a reference table for objects. To store the log, a double-buffering strategy is used to store data with minimal I/O overhead. At runtime, the manager processes automated and manual triggers via its built-in methods, merging redundant event data during each frame update before appending it to the active buffer. By referencing a specific object or event ID, developers can attach state-based metadata such as scores or categorical labels to the property file.

The *EventObject* component is attached to relevant entities in the XR environment that should be tracked by the manager. It defines object metadata, tracked properties, and automates the logging of behavior triggers, such as event functions (e.g., Awake, OnDestroy) and physics interactions (e.g., collisions or triggers).

To prevent frame stutter in the environment, the *EventLogManager* uses an asynchronous StreamWriter on a background thread. Data is written to the disk only when the buffer reaches a threshold, through manual invocation or when the session ends, ensuring minimal interference with the application.

3.1.2 Event Data Enhancement. To complete the holistic perspective of the session, XRFlow uses external data sources and activity translation maps to facilitate interoperability. Activity translation maps, based on a CSV-based key-value structure, provide a means to connect and standardize activity names across different systems.

Using the European Data Format (EDF+) [11], physiological data can be imported and linked to the event log directly. By matching parallel-port-based numeric codes with human-readable activity names, the system can automatically calculate physiological statistics for specific actions within activity instances. For example, by processing an ECG, the tool can determine the heart rate for each individual trial, giving controllers an overview of the user's physical state throughout the session.

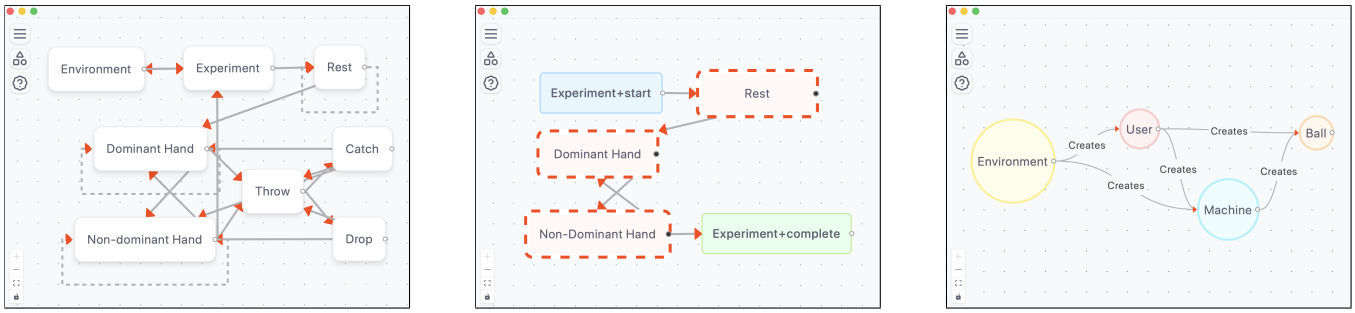
3.1.3 Template Files. Streamlining the analysis of recurring experimental setups requires a method to maintain consistency across multiple sessions. Since data structures and workflows are often identical between participants or trials, XRFlow uses TOML-based template files to capture and reuse specific configurations. These templates store the import operations and settings defined by the controller, creating a reproducible environment for future sessions. By anchoring paths to the project's root directory and using relative file paths, the templates remain portable across different storage locations. If a template is applied to a file outside its root directory, the system prompts the user to migrate existing settings.

3.2 System Implementation

To meet the demands of processing high-frequency XR data, XRFlow is developed as a cross-platform desktop application using the Tauri framework[3]. The backend data management and processing is implemented in Rust, providing the computational efficiency required to parse large event logs and time-series data to generate large graph structures. The user interface shown in Figure 1 is developed in Svelte[4], using Svelte Flow[4] to manage the interactive node-based canvas. This architecture allows the user to explore the process flow directly through a flexible interface, which supports manual node manipulation, panning, and automated layout algorithms to assist in data exploration. A statistical side panel complements the canvas, displaying information based on the currently selected combination of elements.

[3]https://tauri.app/

[4]Svelte: https://svelte.dev/, Svelte Flow: https://svelteflow.dev/



(a) Event-based View **(b) Lifecycle-grouped View** **(c) Object-centric View**

Figure 2: Multi-perspective visualizations of a single XR session.

We emphasize analytical efficiency, allowing users to quickly move from raw data to meaningful conclusions. The application features a drag-and-drop interface for log ingestion, automatically indexing the data to populate the views. To maintain high responsiveness, the system employs a lazy-evaluation strategy supported by an internal caching layer. Rather than pre-computing all data at ingestion, XRFlow calculates view-specific visualizations on-demand. This minimizes inter-process communication overhead between Rust and TypeScript, while caching frequently accessed statistical computations to create a smooth experience.

3.3 Visualizations and Insights

Transitioning from imported raw data, XRFlow helps researchers explore XR sessions through several specialized visualization modes. These perspectives allow for different ways of looking at the same data, ranging from a simple timeline to object-focused history.

3.3.1 Event-based View. At the most basic level, the event-based view generates a directly-follows graph based on the chronology of the event log [17]. The model establishes nodes for each activity and draws an arc from one activity to the next if they occurred sequentially in the log. This generates a global perspective of the entire XR session, mapping out the full sequence of all actions.

For example, the sequence in Table 1 shows a self-loop on the *Rest* node, which then linearly progresses the *Dominant Hand* condition. Although the table only captures a single instance, the full log shows the *Throw* and *Catch* nodes sharing a bidirectional relationship, as the participant repeats these actions many times within the condition. This mapping, shown in Figure 2a, allows controllers to see the connections of the entire experiment.

Short or simple experiments may find this global view sufficient to capture the necessary context. However, while the global view helps give a general idea of the entire process, it ignores the local context of sub-processes. It hides important details, such as whether certain activities only happen under specific experimental conditions or whether they are tied to a particular object in the scene. Furthermore, high data complexity risks creating "spaghetti models", where an excessive number of connections makes the view difficult to analyze.

3.3.2 Lifecycle-grouped View. This perspective transitions the data representation from a global event stream to localized sub-traces. The *Start* and *Complete* markers within the lifecycle column delineate the log into local sub-logs, defining the boundaries of each lifecycle-defined activity. To ensure data integrity, the system attempts to pair these markers using their unique object IDs. If an ID is missing or if multiple identical IDs appear in parallel lifecycles, the system employs a First-In-First-Out (FIFO) fallback to close the sub trace. Any events occurring between these markers are linked as children of that specific sub-trace. This structure compresses the event log into a hierarchical tree, where sub-traces are nested within one another, and instantaneous events serve as the leaves.

The Lifecycle view collapses each sub-trace to show only the current level of the hierarchy. The internal nodes are visually distinguished by thick dashed borders, while the leaves match the event-based view. By selecting an internal node, the controller can navigate the tree to observe different granularity levels of the event data. Once selected, the internal node acts as the source and sink of the current level, differentiated by color and including their lifecycle state. This interaction effectively isolates the sub-traces from the global environment, allowing for the analysis of specific task iterations without the interference of unrelated session data.

Figure 2b shows the top-level hierarchy of the running example, showcasing the primary phases of the experiment. In this view, lifecycle-defined activities, such as the resting state or the tasks involving the dominant and non-dominant hands, are encapsulated within their own sub-traces. This abstraction hides the internal complexity of each activity, navigable via double-click to inspect the isolated events within.

Localizing the traces in this manner allows for analysis of behavior relevant only to that specific lifecycle-defined activity. For instance, if a participant successfully caught every ball with their dominant hand, but dropped several with their non-dominant hand, the sub-traces would reflect these differing performance patterns.

While this view clarifies the global event structure and its local components, it does not yet reveal how individual objects interact with each other.

3.3.3 Object-centric View. The object-centric view visualizes the relational dependencies and interaction patterns between entities within the XR environment. Rather than following a chronological flow, this perspective utilizes implicit graph-based feature extraction to map connections [2] based on shared events and lifecycle overlaps. By identifying events that co-exist across multiple objects,

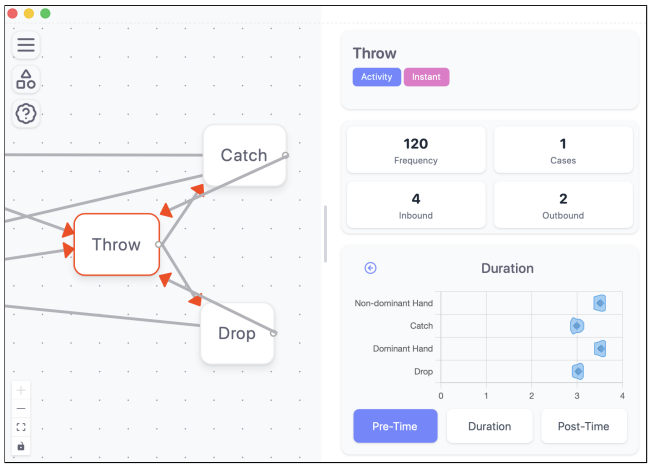


Figure 3: Pre-Time interval distributions across source events.

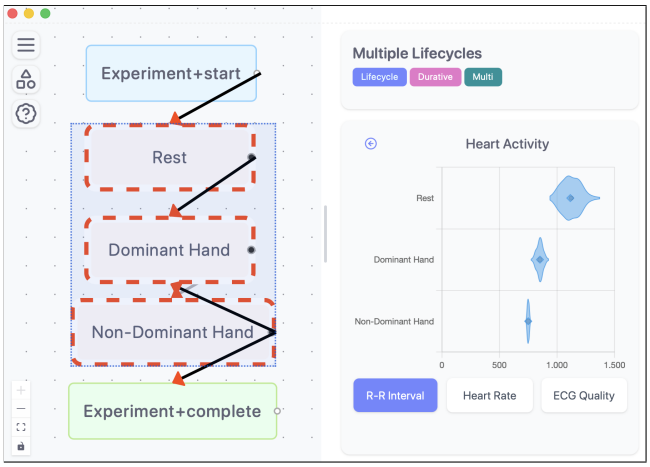


Figure 4: R-R interval distributions across activity lifecycles.

the system extracts relational dynamics as feature maps, without explicit manual tagging.

Figure 2c illustrates a subset of relations within the running example. The object creation feature map reveals a causal hierarchy where the environment first initializes, followed by users and their machines, which then spawn ball objects. These maps are derived from event co-occurrence. For example, a creation pattern is formed between the *Environment* and *User*, because the former precedes and shares the initial event with the latter. Other variants include the interaction map, representing the participation of shared events, and the continuation map, which identifies temporal handoffs where one object terminates as another appears [2].

Although the view primarily represents a node-link graph with bidirectional edges, it implicitly conveys temporal progression through these directional creation relationships. To connect the high-level object map with event data, controllers can double-click object nodes to explore the isolated sub-trace of the events related to that object. This allows data controllers to verify the causal flow of the experiment at a glance while maintaining the ability to explore further into specific object histories.

3.3.4 Statistical Panel. The statistical panel serves as a context-sensitive interface for the analysis of session elements applicable in all views. To preserve screen real estate for the process model, the panel remains hidden until a valid selection is made. The interface is split into two sections. The upper section displays standardized metadata and graph-based metrics, such as frequency, input/output counts, and case numbers. In contrast, the lower section dynamically populates available data dimensions, such as categorical, numerical, time-based properties and physiological streams.

Figure 1 shows that when selecting an activity such as *Catch*, the interface displays graph-based metrics and relevant properties. Selecting a specific property replaces the category list with a violin plot visualizing the distribution of that attribute, as seen in Figures 3 and 4. In addition, controllers can simultaneously select multiple items of the same type to compare their properties. When aggregating events, lifecycles or objects, the panel identifies the intersection of shared numerical properties between the items. For instance, if two objects share an *Age* attribute, the violin plot renders a unified distribution of all combined data points, while properties unique to only one of the objects are automatically hidden to maintain a focused analysis.

4 Practical Application

4.1 Technical Debugging

To evaluate the effectiveness in identifying logic flaws, we simulated a timing error within the ball-catching task. Temporal jitter is typically added between trials to prevent the participant from predicting the stimulus. Although the design specified a 3-second delay (±20%) between *Catch* and *Throw*, a bug was introduced that reduced this jitter to only ±5%. Visually, this error is nearly indiscernible during a live or recorded session, as the average delay remains approximately three seconds.

The metrics are accessed by selecting the *Throw* activity in the event-based view and opening the duration-based properties panel section. By isolating the *Throw*, a developer can examine the duration of the transition or the latency preceding the *Catch* activity. As shown in Figure 3, this panel generates a violin plot of these durations. The resulting slim distribution is concentrated in a 300 ms window rather than the intended 1.2 s span. Therefore, the option to visualize the distributions of properties attached to events and objects in this manner enables the detection of errors that are otherwise hidden in aggregate averages.

4.2 Multimodal Analysis

Beyond identifying technical discrepancies in event logic, the framework enhances the review of physiological data between experimental phases. In the running example, an ECG was recorded to analyze the physiological differences between using the dominant and non-dominant hand during the ball catching task. A baseline resting period was also included for comparison.

The system aligns these data streams without external synchronization protocols by matching discrete event markers found in both the Unity log and the physiological dataset. To resolve naming conflicts, a pre-made translation map converts numeric event

codes from serial connections into the activity names used in the event log. Once aligned, the data becomes available to be processed through the windows defined by the lifecycle-defined activities. Since these activities encompass a duration, they define the temporal boundaries used for the feature extraction and analysis.

Selecting a lifecycle activity allows researchers to view physiological insights derived from external time-series data. If multiple lifecycles are selected, the interface displays vertically-stacked violin plots for each condition as in Figure 4. In this example, the layout enables a direct comparison of R-R interval distributions, i.e., the time between heartbeats, within a single session.

The resulting plots indicate that the resting phase was the most relaxed state, containing the highest time between beats at around 1100ms. In contrast, catching with the non-dominant hand shows a shorter interval compared to the dominant hand, potentially indicating a greater task load. This automated segmentation provides a high-level overview of data integrity and experimental effects immediately after a session.

5 Discussion and Future Work

The development of XRFlow was guided by a philosophy of immediate feedback and accessible integration. Unlike traditional analysis workflows that require manual post-processing, this system enables controllers to explore a live, interactive overview through direct interaction. Using an object-centric data model, we move away from flat event logs and toward a structured representation of lifecycles. This approach aligns with the needs of both developers identifying logic discrepancies and researchers validating experimental designs.

XRFlow generalizes the multimodal data assessment process by maintaining a separation between event logic and time-series analysis. The architectural modularity ensures that the framework is highly extensible from the perspective of process mining and time-series data analysis. A researcher can implement new feature extraction methods for new physiological hardware sensors, e.g., electrodermal activity or eye tracking, without needing to understand the underlying process mining algorithms to split the data. Conversely, a developer could update the event based logic without interfering with the physiological data pipelines.

More importantly, this technical decoupling directly enhances the user experience and overall workflow efficiency. By automating the alignment and segmentation of separate data streams, the system facilitates ad-hoc validation of technical logic and experimental hypotheses. These immediate responses help shorten the iteration cycle between XR application design and data verification. Overall, XRFlow acts as a flexible foundation for multidimensional computations across a wide variety of XR projects, ensuring that the tool remains useful as both sensors and research questions evolve.

However, significant technical and workflow limitations and trade-offs remain. Currently, the interface is optimized for single-session overviews. Although multiple sessions can be imported, the interface does not yet feature a dedicated object-comparison view. Consequently, the event data is aggregated from multiple imported sessions without differentiating between object or user. For a data controller, this makes side-by-side comparisons across objects difficult without external processing.

The quality of automated analyses also depends on how the event-logging system is integrated. If the XR application fails to define clear lifecycles and events with relevant objects, the system cannot faithfully represent the session. Consequently, the overall utility as an analysis tool is predicated on a developer's understanding and adherence to a structured, object-centric logging design.

Finally, the automated nature of the statistical computations creates a black box effect with regard to signal quality. Data controllers cannot observe raw data or exact processing steps without viewing the source code, trusting that the automated steps work properly. For example, an ECG may contain large muscle artifacts, adding large artifacts that cannot be excluded, significantly affecting the results. Therefore, while the system's overview approach is sufficient for identifying high-level trends and validating experimental effects, it does not replace the need for rigorous analysis pipelines for formal statistical reporting.

The current version of XRFlow sets the foundation for quick analyses, but still lacks some useful features. A primary goal for future development is to implement real-time session monitoring using the Lab Streaming Layer (LSL) [13] in addition to the current post-hoc analysis. This would enable the live generation of the visual process model, allowing data controllers to monitor the participant's progression through the experimental logic in real-time. By observing the live creation of the process model alongside streaming physiological distributions, validation could then occur in the moment while the participant is still in the headset, identifying logic errors or physiological anomalies as they occur.

In contrast to the tools described in Section 2, our roadmap includes a simplified replay system designed to enable direct session navigation through events and objects. This view is intended as a secondary resource for contextual validation rather than a primary analysis tool. Selecting events or objects would trigger an automatic jump to the corresponding temporal state, eliminating manual scrubbing. The lightweight approach focuses on providing a general overview while avoiding the technical overhead of loading project-specific assets. The user-represented objects (headset and controllers) could then be rendered with overlaid physiological data. For example, a heart icon above the HMD could dynamically display heart rate, exposing immediate spatial context for physiological spikes. By identifying movement-induced noise such as sudden reaching motion, this context could prevent misinterpretation of data and provide more reliable behavioral insights.

6 Conclusion

This paper introduced XRFlow , a tool that streamlines the analysis of multimodal XR data through an object-centric lifecycle model. By automating the synchronization of physiological signals with interactive events, the tool provides an efficient validation loop that was previously bottlenecked by setting up manual processing pipelines. Although the system's accuracy is predicated on a structured event-logging design, it provides a scalable foundation for real-time experimental monitoring and behavioral insight. In practice, XRFlow lowers the technical barrier for data controllers, transforming recorded XR session events and physiological data into an accessible visual map of the experience.

Acknowledgments

This study has been supported by the Luxembourg National Research Fund (FNR) under grant number C22/BM/17121206 (VR BIAS).

References

[1] Annalisa Appice, Pasquale Ardimento, Donato Malerba, Giuseppe Modugno, Diego Marra, and Marco Mottola. 2020. Training in a Virtual Learning Environment: A Process Mining Approach. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. 1–8. doi:10.1109/EAIS48028.2020.9122760 ISSN: 2473-4691.

[2] Alessandro Berti, Johannes Herforth, Mahnaz Sadat Qafari, and Wil M. P. van der Aalst. 2024. Graph-based feature extraction on object-centric event logs. *International Journal of Data Science and Analytics* 18, 2 (Aug. 2024), 139–155. doi:10.1007/s41060-023-00428-2

[3] Alessandro Berti and Wil M. P. van der Aalst. 2023. OC-PM: analyzing object-centric event logs and process models. *International Journal on Software Tools for Technology Transfer* 25, 1 (Feb. 2023), 1–17. doi:10.1007/s10009-022-00668-w

[4] Juanita Caballero Villalobos, Simon James Jensen, and Hugo A. López-Acosta. 2024. 3DCR: A Tool for Immersive Process Mining: 6th International Conference on Process Mining. *Proceedings of the 6th International Conference on Process Mining* (2024). Publisher: CEUR-WS.

[5] Jonas Deuchler, Wladimir Hettmann, Daniel Hepperle, and Matthias Wölfel. 2023. Streamlining Physiological Observations in Immersive Virtual Reality Studies with the Virtual Reality Scientific Toolkit. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 485–488. doi:10.1109/VRW58643.2023.00105

[6] Claudia Diamantini, Laura Genga, and Domenico Potena. 2016. Behavioral process mining for unstructured processes. *Journal of Intelligent Information Systems* 47, 1 (Aug. 2016), 5–32. doi:10.1007/s10844-016-0394-7

[7] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. 2021. OCEL: A Standard for Object-Centric Event Logs. In *New Trends in Database and Information Systems*, Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, Raimundas Matulevičius, Ahmed Awad, Matthias Weidlich, Mirjana Ivanović, and Olaf Hartig (Eds.). Springer International Publishing, Cham, 169–175. doi:10.1007/978-3-030-85082-1_16

[8] Pablo Gomez, Autumn R Anderson, and Ana Baciero. 2017. Lessons for psychology laboratories from industrial laboratories. *Research Ethics* 13, 3-4 (July 2017), 155–160. doi:10.1177/1747016117693827

[9] Sebastian Hubenschmid, Jonathan Wieland, Daniel Immanuel Fink, Andrea Batch, Johannes Zagermann, Niklas Elmqvist, and Harald Reiterer. 2022. ReLive: Bridging In-Situ and Ex-Situ Visual Analytics for Analyzing Mixed Reality User Studies. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–20. doi:10.1145/3491102.3517550

[10] Charles Javerliat, Sophie Villenave, Pierre Raimbaud, and Guillaume Lavoué. 2024. PLUME: Record, Replay, Analyze and Share User Behavior in 6DoF XR Experiences. *IEEE Transactions on Visualization and Computer Graphics* 30, 5 (May 2024), 2087–2097. doi:10.1109/TVCG.2024.3372107

[11] Bob Kemp and Jesus Olivan. 2003. European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. *Clinical Neurophysiology* 114, 9 (Sept. 2003), 1755–1761. doi:10.1016/S1388-2457(03)00123-8

[12] Yoonsang Kim, Zainab Aamir, Mithilesh Singh, Saeed Boorboor, Klaus Mueller, and Arie E. Kaufman. 2025. Explainable XR: Understanding User Behaviors of XR Environments Using LLM-Assisted Analytics Framework. *IEEE Transactions on Visualization and Computer Graphics* 31, 5 (May 2025), 2756–2766. doi:10.1109/TVCG.2025.3549537

[13] Christian Kothe, Seyed Yahya Shirazi, Tristan Stenner, David Medine, Chadwick Boulay, Matthew I. Grivich, Fiorenzo Artoni, Tim Mullen, Arnaud Delorme, and Scott Makeig. 2025. The Lab Streaming Layer for Synchronized Multimodal Recording. *Imaging Neuroscience* 3 (2025), IMAG.a.136. doi:10.1162/IMAG.a.136 Publisher: MIT Press.

[14] Roy Oberhauser. 2022. VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality. In *Proceedings of the Fourteenth International Conference on Information, Process, and Knowledge Management*. Elektronik und Informatik, 75 – 80. https://www.thinkmind.org/index.php?view=article&articleid=eknow_2022_3_40_60017

[15] Vladimir A. Rubin, Alexey A. Mitsyuk, Irina A. Lomazova, and Wil M. P. van der Aalst. 2014. Process mining can be applied to software too!. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. Association for Computing Machinery, New York, NY, USA, 1–8. doi:10.1145/2652524.2652583

[16] Andreas Schoknecht, Tom Thaler, Peter Fettke, Andreas Oberweis, and Ralf Laue. 2017. Similarity of Business Process Models—A State-of-the-Art Analysis. *ACM Comput. Surv.* 50, 4 (Aug. 2017), 52:1–52:33. doi:10.1145/3092694

[17] Wil Van Der Aalst. 2016. *Process Mining*. Springer, Berlin, Heidelberg. doi:10.1007/978-3-662-49851-4

[18] Wil M. P. Van Der Aalst. 2023. Object-Centric Process Mining: An Introduction. In *Formal Methods for an Informal World*, Antonio Cerone (Ed.). Vol. 13490. Springer International Publishing, Cham, 73–105. doi:10.1007/978-3-031-43678-9_3 Series Title: Lecture Notes in Computer Science.

[19] Sebastiaan J. van Zelst, Felix Mannhardt, Massimiliano de Leoni, and Agnes Koschmider. 2021. Event abstraction in process mining: literature review and taxonomy. *Granular Computing* 6, 3 (July 2021), 719–736. doi:10.1007/s41066-020-00226-2

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009