Framework for Extracting Real-World Object-Centric Event Logs from Game Data

Lukas Liss[1(✉)], Nico Elbert[2], Christoph M. Flath[2], and Wil M. P. van der Aalst[1]

[1] RWTH Aachen University, Aachen, Germany
{liss,wvdaalst}@pads.rwth-aachen.de
[2] Julius Maximillians Universität Würzburg, Würzburg, Germany
{nico.elbert,christoph.flath}@uni-wuerzburg.de

Abstract. In recent years, process mining has shifted towards an object-centric perspective on processes, considering interacting sub-processes that operate on objects from different types. Since businesses are hesitant to publicly share such event data that may expose business internals, there is a lack of public real-world object-centric event logs. We propose a novel approach to use publicly accessible game data as a large-scale data source for object-centric event logs. The contributions of this paper include (1) a framework to extract object-centric event logs from real-time strategy game data, (2) an application of that framework for the game Age of Empires II, (3) a published Python library to automatically transform Age of Empire gameplays into an object-centric event log, (4) a publicly accessible object-centric event log extracted from 325,398 gameplays, and (5) an evaluation of the published *aoe2pm* library. The evaluation shows that the extracted event logs can be used by state-of-the-art applications to generate relevant behavior insights that require an object-centric perspective. The size and attribute richness of the object-centric event log motivate future research questions in the field of object-centric process mining.

Keywords: Object-centric Event Data · Data Extraction · Process Mining

1 Introduction

Publicly accessible datasets are essential for process mining research [20]. Process mining analyzes event data to generate insights into processes. Typical process mining pipelines span discovery, conformance checking, and enhancement. Event data are an essential input for most process mining algorithms in each of these steps. The recent increased interest in machine learning applications in the field of process mining amplifies the need for even larger datasets further [5,11].

In recent years, a new perspective on processes gained traction in process mining research, namely object-centric process mining [22]. In object-centric

A. Delgado and T. Slaats (Eds.): ICPM 2024 Workshops, LNBIP 533, pp. 363–375, 2025.
https://doi.org/10.1007/978-3-031-82225-4_27

process mining a process can consist of interacting sub processes that operate on multiple objects of different types. In a hospital process, for example, an object-centric process can contain the behavior of the patients as well as the behavior of the doctors. Where some events like an *operation* involve both doctor and patient objects, other events like *sit in waiting room* may only involve objects of type patient. Traditional process mining assumes a single case identifier, limiting the perspective to one entity, for example, the patient or the doctor.

To support a holistic object-centric perspective on processes, new data formats like OCEL 2.0 [3] or event graphs [9] for object-centric event logs have been proposed. Because of this holistic view, companies may hesitate to publish real-world object-centric event logs possibly revealing process internals. Due to this and the novelty of object-centric event log formats, real-world object-centric event logs are very limited.

As further elaborated in Subsect. 3.2, real-time strategy (RTS) games share many characteristics that are essential for business processes like goal orientation, competitive behavior, efficient resource utilization, striving for automation, and human errors [8]. Thus, we advocate the use of RTS game data to close the gap of missing object-centric event data. This game category includes popular games such as *Starcraft 2*, AOE2, *Northgard*, or *Dune: Spice Wars*. Most games allow the export of gameplays, and there exist special community-driven websites where players publish their game data to compare themselves with other players.

Individuals are more willing to share their process data publicly than companies are, especially in the context of games, which is why multiple research domains like machine learning utilized game data as a source for large-scale datasets and scalability research [7]. To the best of our knowledge, game data has not yet been used to create object-centric event logs.

Since process mining is especially interested in the dynamic behavior between people, objects, and systems, publicly accessible object-centric event logs with real human involvement are essential for developing and evaluating process mining methods. Event logs from RTS game data can contribute to this.

This paper presents five contributions to enable the usage of game data in object-centric process mining. First, we propose a framework to extract object-centric event logs from RTS games. Second, we apply the framework to the game *Age of Empires II* (AOE2). Third, we provide a publicly accessible python library (*aoe2pm*[1]) that transform exported AOE2 game data to object-centric event logs. Fourth, we published an object-centric event log[2] extracted from 325,398 AOE2 matches sourced from community websites. Fifth, we perform a quantitative and qualitative evaluation of the framework using *aoe2pm*, showing that the object-centric event logs are compatible with academic and commercial state-of-the-art object-centric algorithms, resulting in relevant process insights.

[1] https://github.com/nicoelbert/aoe2PM/releases/tag/v1.0.1.
[2] https://doi.org/10.5281/zenodo.11506365.

2 Related Work

Process mining analyzes event data to provide process-related insights. This spans process discovery, conformance checking, and enhancement [20]. The starting point for most process mining approaches is an event log that captures the events happening in the context of a given process. Traditional process mining assumes a single case identifier for a process, limiting the perspective to the chosen case identifier. Real-world processes tend to consist of multiple sub process that interact with each other and involve multiple object from various types. To represent these interacting multi-object processes, process mining shifted towards object-centric process mining in recent years [22]. Object-centric process mining allows events to involve multiple objects of different types. There are object-centric event formats like event graphs [9] and OCEL 2.0 [3].

Currently, there are two types of object-centric event data extractors. Some work on relational databases commonly used in ERP systems [16,23], and others take traditional event logs and transform them into object-centric event logs [18]. Both approaches do not work on game data logs. Also, both approaches are not designed to add the relevant game logic-based events, which are missing in the gameplay exports as described in Subsect. 3.2. Thus, even when transforming the input format, resulting event logs would be limited to player-based events only. To overcome this limitation, we propose a framework to extract object-centric event data from real-time strategy (RTS) games.

Another approach to creating event logs is to use process simulation [17, 19]. For traditional process mining, there exist multiple approaches that use simulation to create event logs [13,21]. Esser and Fahland [9,10] and Rebmann et al. [18] transformed traditional event logs into object-centric representations. Knopp et al. proposed an approach to create object-centric simulation models [14]. However, simulated event logs can only recreate behavior and do not contain real-world behavior directly.

To overcome the challenges posed by the limited data availability, researchers increasingly leverage competitive online games as an alternative source. Such game data offer large-scale publicly available data that captures human behavior within a naturalistic setting [6]. Driven by personal and intrinsic motivation, players exhibit highly rational decision-making and develop behavioral patterns that are adapted to changing environments. Players engage in long-term skill learning to optimize their performance [12]. Given the setting, they collaborate within intricate organizational and social structures and take on specialized roles essential for individual or collective success. Approaches to utilize this source of process data are so far missing.

3 Background

This section describes object-centric event logs and RTS games that serve as the background for the proposed framework.

3.1 Object Centric Event Data

We use the OCEL 2.0 standard for object-centric event data [3]. Once the process is stored in that format, a variety of tools using that standard can be applied for the analysis [1,4]. Events are activities that happen at a timestamp for a set of objects of different types. $\mathbb{U}_{event}$ is the universe of event identifiers. The universe $\mathbb{U}_{act}$ contains all visible activities. $\mathbb{U}_{type}$ is the universe of all object types. The universe of objects is $\mathbb{U}_{obj}$. Each object has exactly one type associated with it $\pi_{type} : \mathbb{U}_{obj} \rightarrow \mathbb{U}_{type}$. The universe of qualifiers for object-to-object relations is $\mathbb{U}_{qual}$. $\mathbb{U}_{time}$ is the universe of all timestamps. Objects have attributes that can change their values over time. Definition 1 describes the minimal set of features needed for our object-centric event logs. Note that additional OCEL 2.0 features are not considered here.

Definition 1 (Object-Centric Event Log). $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *is an object-centric event log where:*

- $E \subseteq \mathbb{U}_{event}$ *is a set of events,* $O \subseteq \mathbb{U}_{obj}$ *is a set of objects,*
- $OT = \{\pi_{type}(o) | o \in O\}$ *is a set of object types,*
- $O2O \subseteq \{(o_1, o_2, q) | q \in \mathbb{U}_{qual} \wedge o_1, o_2 \in O \wedge o_1 \neq o_2\}$ *is the set of qualified object to object relations,*
- $\pi_{act} : E \rightarrow \mathbb{U}_{act}$ *maps each event to an activity,*
- $\pi_{obj} : E \rightarrow \mathcal{P}(\mathbb{U}_{obj}) \setminus \{\emptyset\}$ *maps each event to at least one object,*
- $\pi_{time} : E \rightarrow \mathbb{U}_{time}$ *maps each event to a timestamp*

3.2 Real-Time Strategy Games

In this subsection, we show why the structure of RTS games naturally facilitates in object centric processes and discuss to what extent RTS games can serve as an approximation for business processes.

In their complexity, RTS interactions are comparable to many real-world problems [2] and allow the observation of human behavior in a naturalistic environment over extended time periods [6]. Business processes are goal-orientated sequences of actions that utilize resources to reach their goal. They often strive for continuous improvement under the constraints of compliance with regulatory rules [8]. This is similar in RTS games, where players must organize and build a micro economy with the goal to outperform their opponents. They have a finite set of actions they can strategically perform and the conditions for winning are well-defined. Players are bound to the rules of the games, but driven by competitive motivation [12], they strive to optimize their performance and decision-making and thus their process. Therefore, similar KPIs are relevant for business processes and RTS games. As stated in Sect. 1, the waiting time of resources and the degree of automation are examples of that. In addition, both business processes and RTS games are based on human behavior and thus incorporate for example human errors. This makes them interesting to investigate from a behavioral point of view.

RTS games have a common structure of game objects and event types, shown in Fig. 1. Each match consists of multiple players that orchestrate working units to create structures and collect natural resources. A session is connected to all events happening on a player's computer in one match. Players create and manage their micro organization of structures and units with the goal of producing more resources than their opponents. The process of creating and managing this micro organization involves multiple objects of different types. Players strive to analyze and optimize this process to outperform their opponents. Popular games following this RTS game structure include, for example, *Starcraft 2*, AOE2, *Northgard*, or *Dune* with millions of players. Unlike turn-based games like chess, RTS games operate on a continuous time granularity [15]. There are player-based events that are triggered by the player and game logic-based events that are triggered by the game. Both types of events can contain multiple game objects of different types. For example, one player can use three units to build a structure. To represent these object interactions and avoid information loss due to convergence, divergence, or deficiency, an object-centric process notion is required [22].

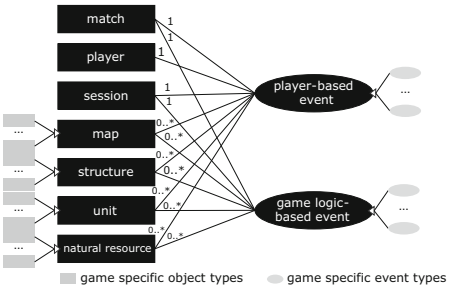

Fig. 1. Object types (boxes) and event types (ellipses) in RTS games.

3.3 Game Export Format

Each match is exported separately in a so called compressed action log file. To save memory, this file only contains player-based events since game logic-based events can be recreated by the game engine. The file contains the user inputs and relates them to involved objects. Each RTS game with an export function must export at least this information to be able to replay the game. Figure 2 shows an example exported compressed action log for AOE2 showing the user input. The event and object information from Fig. 2 are structured and summarized in Table 1.

```
Input(timestamp='0:0:30', type='Queue', param='Villager', payload={'building_id': 70,
'object_ids': [1001, 1002, 1003], 'amount': '3'}, player=P1, position=(x=30, y=50))

Input(timestamp='0:1:05', type='Build', param='House', payload={'object_ids': [1001,
1004], 'building': 'House'}, player=P1, position=(x=45, y=60))

Input(timestamp='0:1:20', type='Gather Point', param='Wood', payload={}, player=P1,
position=(x=25, y=55))
```

Fig. 2. Example of an exported compressed action log from AOE2.

Table 1. Example of an exported compressed action log from AOE2.

Player ID	Timestamp	Action	Object ID	Position
1	00:00:30	Queue Villager	1001, 1002, 1003	(30, 50)
1	00:01:05	Build House	1004, 1001	(45, 60)
1	00:01:20	Set Gather Point		(25, 55)

An example of such a compressed action log for the game AOE2 can be seen in Table 1. There, player 1 first queues three villagers, who are the workforce resource in AOE2, then builds a house with one of the villagers and sets a gather point. These gather points in AOE2 are a way to automatically assign tasks to idle villagers to go and gather wood. To investigate relevant process KPIs like waiting time of resources or degree of automation, accessing only the player related events is not enough. For example, in Table 1, the waiting time of a villager before building a house seems to be 35 s. However, resources like villagers are not immediately available after being queued. Villagers are created (spawned) one by one after 30 s each by the game engine. Since that event is triggered by the game engine and not the player, it is not recorded in the compressed action log. The correct waiting time for the resource villager before building the house would be 5 s, but to be able to correctly detect that, the game logic-based events are also required. Table 2 shows the desired object-centric event log for the example in Table 1 with the game logic-based events highlighted by colors. Without the game logic-based events, many relevant business process KPIs are misleading. Another example of that is the degree of automation. Without the game logic-based events, the degree of automation would always be 0 because we only see events that involve the player. Thus, players who use automation mechanisms like gathering points cannot be distinguished from players who do not. Therefore, extending the compressed action logs and then transforming them to an object-centric event data format is necessary to enable insightful process mining on game data.

Table 2. Desired object-centric event log for the example game recording in Table 1. Game logic-based events are colored based on our framework's type of game rule used to add them (orange = construction time, blue = unit queries, green = automation).

			object types			
event id	**activity**	**timestamp**	**player**	**villager**	**house**	**...**
e1	Queue Villager	17:00:30	p01	v01, v02, v03		
e2	Spawn Villager	17:01:00		v01		
e3	Command Build House	17:01:05	p01	v01	h01	
e4	Set Gather Point	17:01:20	p01			
e5	Spawn Villager	17:01:30		v02		
e6	Gather Wood	17:01:31		v02		
e7	Spawn Villager	17:02:00		v03		
e8	Gather Wood	17:02:01		v03		
e9	Complete House	17:06:05		v01	h01	

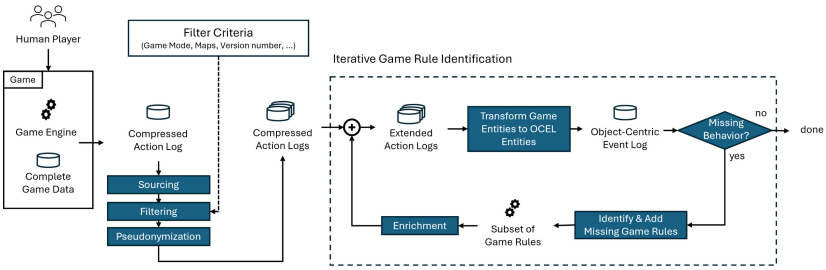

Fig. 3. Overview of the framework transforming RTS game data to object-centric logs.

4 Framework

This section outlines the framework for extracting object-centric event logs from RTS games, exporting gameplays in a compressed action log format that only contains player-based events. Figure 3 provides an overview.

Gameplay and Exporting: RTS game data originates from player interactions, resulting in compressed action logs, capturing real-world human behavior that shares key characteristics with business processes. RTS games export game interactions as compressed action logs that only contain player-based events to save memory as described in Subsect. 3.3.

Sourcing: To transform exported compressed action logs into an object-centric event log, game exports must first be collected. This sourcing step can be done through user studies, where selected participants play a controlled game setting, minimizing external influences but limiting event log size. Alternatively, pre-existing large-scale data from online communities, accessible via web scraping, can be used. There exist multiple community websites where RTS games upload game exports automatically or players upload them to compare themselves with others. Utilizing publicly available data enables the creation of large-scale event logs, capturing diverse player behaviors and strategies. The data source should be clearly indicated in the log metadata [6].

Filtering: Games vary in versions and settings, necessitating filtering of exported data to ensure consistent settings. Filter criteria should be documented.

Pseudonymization: In line with responsible data science, personal data in game replays must be pseudonymized, including clear text names and tags.

Enrichment: After filtering and pseudonymization, game data is enriched with inferred game logic-based events. This process involves iteratively adding game rules until the desired level of detail is achieved in the object-centric event log. Game rules are functions that extend action logs with game logic-based events such that the returned extended action log contains more events than before. The added game-logic based events are based on the player-based events. We

identified three types of game rules: construction time rules, unit queue rules, and automation method rules. Additionally, game-specific extra rules can exist. Construction rules add events for structures that do not involve the player, for example when the construction is finished by the working units (see the orange events in Table 2). Units can be ordered in batches but are created (spawned) one after each other based on a production queue. These spawning events are added by unit queue rules that simulate the queuing (see the blue event in Table 2). RTS games offer automation methods that for example allow to assign idling units tasks automatically. For example, in AOE2, by setting gather points, which automatically sends workers to gather wood when they are free (see the green events in Table 2). The extended action logs are then transformed to an object-centric event log as described in the next step. If the resulting log contains all desired behavior, the framework ends, otherwise missing game rules are identified and added in each iteration until all desired behavior is covered.

Identify and Extract OCEL-Specific Entities and Relations: This step takes multiple action logs and returns an object-centric event log. This involves identifying events, objects, and their relationships in the game data, including event-to-object relations and object-to-object relations. The final output is an OCEL 2.0 format log, compatible with tools like pm4py [4] and ocpa [1]. The game-specific identification utilizes domain knowledge. For example, in AOE2, the activity name is split in the compressed action log. The first stored user input in Fig. 2 shows type *Queue* and param *Villager* which combined gives the activity name. Since identifiers for the events and objects are only unique within one action log, new globally unique identifiers are assigned to merge them into the object-centric event logs events E and objects O. The activity, object, and time mappings π_{act}, π_{obj}, and π_{time} for the events are adapted to the global identifiers. This mapping allows combining action logs into one cohesive OCEL 2.0 event log capturing all matches.

5 Implementation and Resulting Log for AOE2

This section describes the implementation of the framework for the game AOE2 and the resulting object-centric event log.

We applied our framework to the game AOE2, which resulted in a publicly accessible Python library *aoe2pm*. We chose AOE2 because it is one of the most established RTS games with well-organized community websites with publicly accessible compressed action logs (called .aoe2record files in AOE2). The library development followed the iterative framework presented in Fig. 4 to identify a suitable set of game rules for an authentic game state representation. For example, the initial iteration revealed missing behaviors for the working unit *villagers* as explained in the example in Subsect. 3.3. Thus, an according game rule was added to *aoe2pm* such that related game logic-based events are included in the resulting object-centric event log. The sum of all the added game rules allows *aoe2pm* to create logs that contain the desired game logic-based events that are highlighted by color in Table 2 and are required for meaningful insights.

Using a web scraper, we sourced 325,398 compressed action logs from AOE2 community websites. To keep the matches comparable, we filtered for uniform game versions and match settings, which resulted in 1,000 matches. Using our *aoe2pm* library, we processed these .aoe2record files into an object-centric event log in the OCEL 2.0 format that is publicly accessible. The event log consists of 262,994 events from 494 event types and 40,625 objects from 27 object types. Due to space limitations, we do not cover the concrete event types and their meaning in detail here. But the online description of the publicly accessible object-centric event log contains a detailed list with explanations.

6 Evaluation

First, we evaluate quantitatively the performance and correctness of the framework implementation for AOE2. Showing that *aoe2pm* is suitable for creating large scale event logs that can be used in research and industry tools. Second, we evaluate the usability of the resulting object-centric event log qualitatively. We compare the insights from the object-centric event log resulting from our framework, which contains game logic-based events, with the insights for a pure player-based event log. This shows the qualitative improvement for extracting RTS game logs with our approach over state of the art object-centric extractors that are limited to the directly logged player-based events.

Using the sourced AOE2 compressed game logs, we evaluated the runtime of *aoe2pm*. The runtime grows approximately linear with the number of compressed game log files. The average runtime was 6 s for 1 file, 74 s for 100 files, and 801 s for 1000 files on a machine with an Apple M3 processor and 16GM RAM, tacking the times five times per number of files. We imported the created event logs in pm4py [4] and ocpa [1] to validate the compatibility with state-of-the-art research libraries. As shown in Fig. 4, we uploaded the log to Celonis to validate the compatibility with industry tools using a publicly accessible upload script[3]. This shows that correctly formatted large-scale object-centric event logs can be created with *aoe2pm* on consumer hardware.

For the qualitative evaluation, we focused on two process metrics, namely the waiting time and the degree of automation. For both, we evaluate the importance of the game's logic-based events. An important object-centric process metric is the waiting time of resources before they are used. In AOE2, villagers are a good example of that because players want to utilize them as efficiently as possible to build as many structures like *Houses*, *Barracks*, or *Blacksmiths* as possible. Using discovery in Celonis, we created the object-centric directly follows graph one can see in Fig. 4. It shows the average waiting time of villagers before they start building structures. One can see that the two events before a villager starts building a structure are both game logic-based, namely *Start Queue Villager* and *Complete Queue Villager*. Thus, missing game logic-based events would lead to an overestimation of waiting time before building structures. For the given log, the waiting time of villagers would be overestimated by 4.72 min on average.

[3] https://github.com/Javert899/ocel20-celonis-connector.

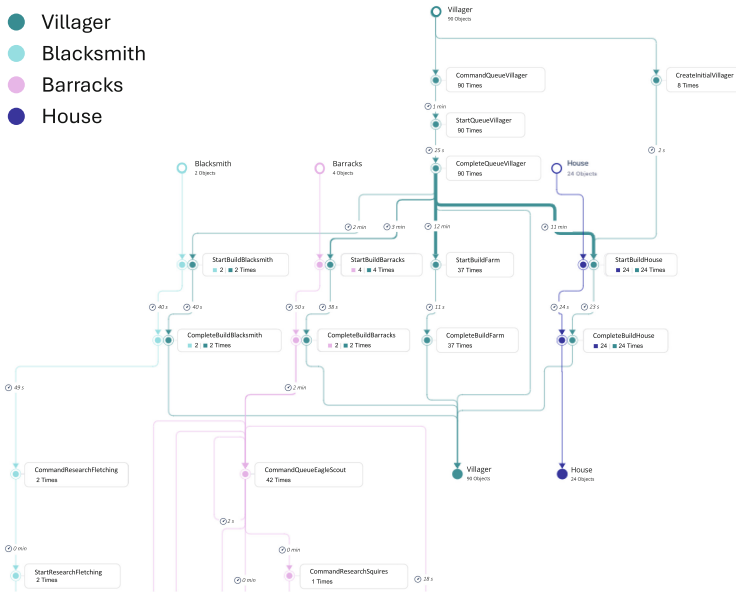

Fig. 4. Screenshot of the object-centric process explorer in Celonis loaded from an object-centric event log created from an AOE2 gameplay export with *aoe2pm*. The graph is filtered for the object types and activities visible in the screenshot. The directly following relations are annotated with the average throughput times.

The degree of automation (see Definition 2) describes how many events happen without player involvement but that belong to the same game session, so the degree of events that are part of the player's economy but do not directly involve the player. This is an important metric for players since the more automated their micro economy functions, the more time they have for advancing and extending their economy.

Definition 2 (Degree of Automation). *Given object-centric event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$*, player* p*, and session* s*, the degree of automation up to timestamp* $t \in \mathbb{U}_{time}$ *is:*

$$\frac{|\{e \in E | s \in \pi_{obj}(e) \wedge p \notin \pi_{obj}(e) \wedge \pi_{time}(e) \leq t\}|}{|\{e \in E | s \in \pi_{obj}(e) \wedge \pi_{time}(e) \leq t\}|}$$

Without the involvement of game logic-based events, which is what current state-of-the-art extractors are limited to, the degree of automation would falsely always be identified as 0%. In the published event log, we showed that the degree of automation rises over time and that players differ in how automated they play, as one can see in Fig. 5. This matches the domain knowledge of how people interact with the game. Making it an interesting metric for user behavior analysis, enabled by the added game logic-based events.

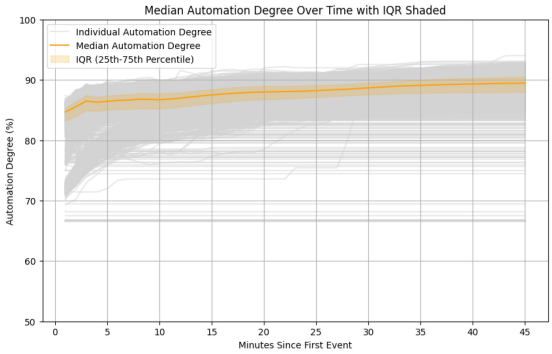

Fig. 5. Degree of automation players reach over the passed game time.

All in all, the qualitative evaluation showed that game logic-infused game logs give indeed better process mining insights than pure player-based ones. The quantitative evaluation shows that large-scale event logs can be created without intensive hardware requirements.

7 Conclusion

In this paper, we proposed a framework to extract object-centric event data from game data and applied it to AOE2, resulting in the Python library *aoe2pm*. We sourced 325,398 AOE2 matches and created a publicly accessible object-centric event log. We performed an evaluation of the framework, demonstrating its ability to create relevant object-centric event logs suitable for object-centric process mining methods like discovery and metric computation. Thus providing a new large-scale source for object-centric event data.

The framework is currently tailored for RTS games and requires adaptation for other game types as future research. Additionally, applying the framework demands domain knowledge, particularly in selecting suitable game rules. Future work includes extending the framework to other games. There is also potential to deepen the object-centric analysis. For instance, there exist community drive best practices for the games, so called build orders which can be used as normative models for object-centric conformance checking. The published event log also motivates reproducibility studies and new research into complex user behaviors.

Acknowledgments. The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project Bridging AI (grant no. 16DHBKI023).

SPONSORED BY THE
Federal Ministry
of Education
and Research

References

1. Adams, J.N., Park, G., van der Aalst, W.M.P.: ocpa: a python library for object-centric process analysis. Softw. Impacts **14**, 100438 (2022)
2. Bayrak, A.E., McComb, C., Cagan, J., Kotovsky, K.: A strategic decision-making architecture toward hybrid teams for dynamic competitive problems. Decis. Support Syst. **144**, 113490 (2021)
3. Berti, A., Koren, I., Adams, J.N., et al.: OCEL (Object-Centric Event Log) 2.0 Specification (2023)
4. Berti, A., van Zelst, S.J., Schuster, D.: PM4Py: a process mining library for python. Softw. Impacts **17**, 100556 (2023)
5. Bozorgi, Z.D., Teinemaa, I., Dumas, M., La Rosa, M., Polyvyanyy, A.: Process mining meets causal machine learning: discovering causal rules from event logs. In: ICPM, pp. 129–136. IEEE (2020)
6. Chen, J., He, S., Yang, X.: Platform loophole exploitation, recovery measures, and user engagement: a quasi-natural experiment in online gaming. Inf. Syst. Res. **35**(4), 1609–1633 (2023)
7. Churchill, D., Buro, M.: Build order optimization in starcraft. In: AAAI, vol. 7, pp. 14–19 (2011)
8. Dumas, M., Rosa, L.M., Mendling, J., Reijers, A.H.: Fundamentals of Business Process Management. Springer (2013)
9. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**(1–2), 109–141 (2021)
10. Fahland, D., Esser, S.: Event graph of bpi challenge 2017. tu.researchdata.dataset (2021). https://doi.org/10.4121/14169584.v1
11. Gal, A., Senderovich, A.: Process minding: closing the big data gap. In: BPM. LNCS, vol. 12168, pp. 3–16. Springer (2020)
12. Huang, Y., Jasin, S., Manchanda, P.: "level up": leveraging skill and engagement to maximize player game-play in online video games. Inf. Syst. Res. **30**(3), 927–947 (2019)
13. Khodyrev, I., Popova, S.: Discrete modeling and simulation of business processes using event logs. In: ICCS. Elsevier (2014)
14. Knopp, B., Pourbafrani, M., van der Aalst, W.M.: Discovering object-centric process simulation models. In: ICPM, pp. 81–88. IEEE (2023)
15. Lara-Cabrera, R., Cotta, C., Fernández-Leiva, A.J.: A review of computational intelligence in RTS games. In: FOCI, pp. 114–121. IEEE (2013)
16. Li, G., de Murillas, E.G.L., de Carvalho, R.M., Van Der Aalst, W.M.: Extracting object-centric event logs to support process mining on databases. In: CAiSE Forum. Springer (2018)
17. Martin, N., Depaire, B., Caris, A.: The use of process mining in a business process simulation context: overview and challenges. In: CIDM, pp. 381–388. IEEE (2014)
18. Rebmann, A., Rehse, J.R., van der Aa, H.: Uncovering object-centric data in classical event logs for the automated transformation from XES to OCEL. In: BPM. LNCS, vol. 13420, pp. 379–396. Springer (2022)
19. Tumay, K.: Business process simulation. In: WSC, pp. 93–98. IEEE Computer Society (1996)
20. van der Aalst, W.M.P.: Process mining. Commun. ACM **55**(8), 76–83 (2012)
21. van der Aalst, W.M.P.: Process mining and simulation: a match made in heaven! In: SummerSim, pp. 4:1–4:12. ACM (2018)

22. van der Aalst, W.M.P.: Object-centric process mining: dealing with divergence and convergence in event data. In: SEFM. Springer (2019)
23. Xiong, J., Xiao, G., Kalayci, T.E., Montali, M., Gu, Z., Calvanese, D.: A virtual knowledge graph based approach for object-centric event logs extraction. In: ICPM Workshops. Springer (2022)