Agent System Event Data: Concepts, Dimensions, Applications

Qingtan Shen[1], Artem Polyvyanyy[1], Nir Lipovetzky[1], and Timotheus Kampik[2]

[1] The University of Melbourne, Victoria 3010, Australia
{qingtan.shen1;artem.polyvyanyy;nir.lipovetzky}@unimelb.edu.au
[2] SAP, Germany
timotheus.kampik@sap.com

Abstract. Event data is a collection of recorded events that capture performed actions and observed states of business processes supported by information systems. It describes the times of event occurrences, event types, event attributes, and process cases of events identified by one or more objects the events relate to. Process mining uses event data to analyze and improve the processes in organizations. These processes are often performed by actors or *agents*, such as employees, resources, and systems, in different roles within organizations. In this paper, we present Agent System Event Data (ASED), a new type of event data that describes business processes as interactions of agents. ASED provides a new scope for analyzing individual agents involved in multiple processes, interactions of agents, and systems of agents that enact the processes. We formalize ASED as a conceptual data model, discuss its dimensional data modeling aspects, and argue that event data, in general, benefits from dimensional representation. We review existing event data types and discuss the complementary nature of existing models and ASED. Finally, we validate ASED by demonstrating its ability to express existing business process compliance rules, significantly expanding the scope of compliance analysis addressed by existing data models.

Keywords: Process mining, event logs, event data, agent system mining

1 Introduction

Process mining is a research discipline at the intersection of data science and business process management [3]. It studies methods and tools to extract insights from event logs recorded during the execution of business processes. An event log is a starting point of every process mining study, consisting of events recorded by information systems during executions of business processes. Each event record can contain information about the triggering activity and process instance, as well as related business objects, activities, resources, and agents. By leveraging event logs, organizations can achieve operational transparency, uncover inefficiencies, and make data-driven decisions [2].

The success of a process mining study hinges on the adept extraction of relevant event data from the information systems that execute a given process. Over the last two decades, the process mining community has meticulously crafted a data model for event data, encapsulating crucial entities and their interrelationships. This conventional

event data model, as described by the XES standard [1], serves as a cornerstone for developing new process mining algorithms and analytics, as well as for preparing event data for use by these techniques. Recently, new process mining paradigms, like Object-Centric Process Mining (OCPM) [4], Robotic Process Mining (RPM) [20], and Agent System Mining (ASM) [28] have emerged. These approaches challenge conventional assumptions about event data, striving to establish effective models that are streamlined in their conceptualization and operationalization while robust in their expressiveness and capacity to facilitate analysis of intricate scenarios.

This paper reviews existing event data models and leverages insights gained to propose a new data model for ASM. ASM, a type of process mining, aims to understand and improve organizations by interpreting them as agent-based systems. In these systems, autonomous agents, including people, machines, and AI, independently perceive their environment, make decisions, and take actions to achieve their goals. These agents interact dynamically and unpredictably with each other and their environment, posing challenges in analyzing their behavior. ASM addresses this by utilizing event data generated by such agent-based systems. Specifically, this paper makes these contributions:

- Introduces the Agent System Event Data (ASED) model, providing a new conceptual framework for event data in Agent System Mining.
- Identifies that ASED has features of a dimensional data model widely used in data warehouses and advocates for the general use of dimensional modeling for event data in process mining.
- Studies the relationship between the ASED model and other event data models in process mining, highlighting the complementary nature of existing data models.
- Validates, by means of a publicly available implementation applied over real-world event data, the ASED model by showing its ability to express a wide range of process compliance rules related to roles, resources, and agents in processes.[3]

The paper proceeds as follows. The next section reviews related work on data models for representing event data in process mining. Then, Section 3 presents event data models that are commonly used in process mining. Next, Section 4 introduces the ASED model and discusses its dimensional nature and its relationship with the other event data models. Section 5 validates ASED by studying its ability to express business process compliance rules. Finally, Section 6 states concluding remarks.

2 Related Work

In this section, we explore various types of process mining, primarily focusing on the process discovery problem studied within the domain. Process discovery involves constructing process models from event data [2].

Event data used as input for conventional process discovery techniques typically includes events with three essential attributes: *activity*, *case identifier*, and *timestamp*. Thus, an event represents an activity triggered at a specific timestamp within a given

[3] The ASED model for the BPIC 2013 event log (10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee) and an implementation of the compliance rules over ASED is available at https://doi.org/10.26188/26868592.v1 and https://doi.org/10.26188/26868442.v2, respectively.

process case. Events sharing the same case identifier, ordered by their timestamps, collectively describe the execution of a process instance, also known as a *trace*. This trace delineates the sequence of activities executed and the events they triggered.

Traditional process mining techniques typically focus on modeling and analyzing processes based on cases captured as a reference to a single object. To overcome this limitation, object-centric process mining (OCPM) methods have been developed to mine and analyze processes involving multiple object types. OCPM can be applied to event logs containing information about events, activities, and the objects and object types associated with each activity [6]. Events processed by OCPM methods can relate to any number of objects; for example, a delivery event may involve both a package and the items inside it simultaneously. The fundamental concept of OCPM is to create a flattened event log for each object type and apply process discovery methods to construct models for each of them [6]. Subsequently, a merging step is applied to combine these object-type models into a comprehensive process model based on the relationships between different objects. Van der Aalst [5] outlined three benefits of OCPM methods. Firstly, data extraction becomes more efficient by focusing on the relationship between events and objects without considering case information. Secondly, relationships between objects can be mined and analyzed from event logs without requiring additional information. Thirdly, object-centric process models can provide visualization of the relationships between objects. OCPM techniques have been applied in various domains. For instance, Li and de Carvalho [21] applied OCPM to social media data to analyze dependencies and relationships between activities and objects. Also, Hobeck and Weber [15] used OCPM on blockchain data to generate an object-centric, directly-follows graph depicting relationships between contracts, markets, and transactions as objects.

Robotic Process Mining (RPM) [20] is defined as a class of techniques to identify candidate routines and use these routines to discover specifications that can be executed by Robotic Process Automation (RPA) bots. The input data, also called the User Interaction (UI) log, is collected from user-driven tasks, which refer to the data generated from interactions between a user and software applications. A UI log is a sequence of events completed by a single user in a single workstation. Each event has a timestamp and is characterized by an event type (e.g., *open email*); no case information is stored in the UI log. An event may have some attributes to store more information. For example, if an event has the type *open file*, then the file name can be stored as an event attribute. The RPM pipeline first preprocesses the input UI log, filters out noise, then identifies suitable event sequences for automation, and finally extracts executable specifications for these sequences. RPM is applied to select sequences of repetitive work that are suitable for automation, to apply RPA, and then to execute scripts for encoding the fine-grained interactions that automate the selected work sequences.

A Multi-Agent System (MAS) is a collection of agents that achieve common or conflicting goals through interactions. Business processes whose participants act independently towards their goals while interacting with other entities, e.g., social interactions over business activities [16], can naturally be modeled as MAS. Agent-Based Modeling (ABM) is a technique that can be applied to describe an organization as a system composed of autonomous agents [28]. Unlike classical process discovery techniques employed to discover a Petri net or BPMN model, agent system mining com-

bines ABM and process mining techniques, aiming to infer the design of a MAS by automatically constructing it from event data. A MAS model often contains two types of components: agent models and models of their interactions. Agent models capture the behaviors of autonomous agents that participate in the system, while interaction models describe patterns of agent interactions. ASM studies ways to construct MAS models from event data, and it comes with at least two benefits compared to classical process discovery methods. Firstly, ASM analyses the behavior of process participants and their interactions [28], which can avoid constructing unreadable models with activities from the same agent scattered across distant parts of the model. Secondly, MAS models constructed by ASM discovery do not necessarily grow in complexity with event data volume growth [28]. Tour et al. [29] introduced the first divide-and-conquer algorithm, called Agent Miner, for automatically discovering a MAS model from event data. The algorithm discovers an interaction model of agents and then each agent's behavior to fit the interaction model.

3 Event Data Models

This section discusses data-driven, event-centric conceptual data models. In the process mining community, Message Sequence Charts (MSCs) that utilize two syntactical constructs, agents and messages, are applied to model message sending and receiving between agents [19], while in the MAS community, several meta-models for agent systems are well-established. Prominently, the AALAADIN meta-model [14] is designed to show the coordination and negotiation schemas of MAS, and the JaCaMo meta-model [9] shows the dependencies, connections, and conceptual mappings between three agent-oriented programming dimensions: environment, agent, and organization. However, these meta-models are not event-centric. In Sections 3.1 to 3.3, we present conventional, object-centric, and user interaction event-centric conceptual data models.

3.1 Conventional Event Data

Conventional event data is a collection of observed and recorded events. Many technologies, including process mining [3] and process querying [24], are applied to event data. Figure 1 shows the conceptual model of conventional event data [6].

An *event* is a record that documents an activity or change within a system or application. It is defined as a mapping from attribute names to attribute values. According to the design in Fig. 1, an event is triggered by exactly one activity, while each activity can trigger multiple events, and it is meaningless to keep records of activities that trigger no events. In addition, each event refers to a process *case* and has a *timestamp*.

An event attribute refers to further specific information associated with an event, beyond the standard event attributes of activity, case, and timestamp, providing various perspectives on the data. Typically, each event encompasses multiple attributes designed to capture different aspects of the event. It is worth noting that some activities may lack event attributes entirely, and presenting irrelevant attributes for such activities serves no purpose. An event can possess any number of event attributes, with each attribute value precisely corresponding to one event and one event attribute. Events may utilize

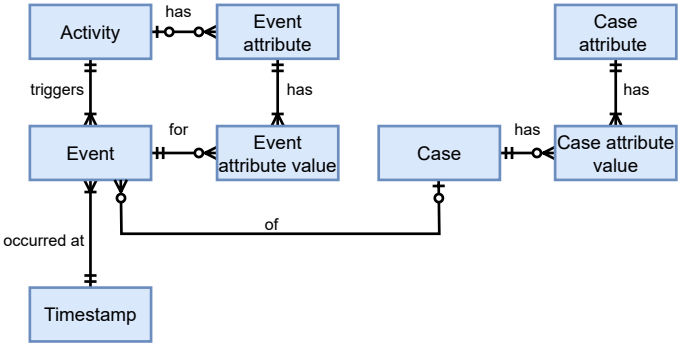

Fig. 1: Conventional event data model [6]

the same attribute to capture information from similar perspectives. However, some events may lack attributes or attribute values. Thus, the relationship between event and event attribute value is mandatory-to-optional. Nevertheless, if an activity with multiple attributes triggers an event, every attribute must have a value corresponding to the event.

A case represents a chronological sequence of events that traverse a particular process. A case can comprise any number of events, and some events may not be associated with any specific case. In conventional event data, it is typically assumed that an event cannot be linked to more than one case. Thus, the relationship between events and cases in the design in Fig. 1 captures this relation as many events can relate to one case. Similarly to events and activities, a case may possess any number of case attributes. Case attributes encapsulate aggregated knowledge derived from all events within the case, enabling the application of querying and other analytical techniques at a higher level of abstraction.

3.2 Object-Centric Event Data

Van der Aalst [6] introduced an Object-Centric Event Data (OCED) model. Similar to the conventional event data model, the OCED model in Fig. 2 has entities and relationships for storing information about events and activities. In addition, OCED supports collecting information about objects; hence, the corresponding entities and relationships are added to the OCED meta-model. Every object has one object type, e.g., object O_1 can be of type package, while object O_2 can be of type item. An event can relate to any number of objects, and an object can relate to any number of events. For example, objects O_1 and O_2 can be involved in the same delivery event. Similarly, object O_1 can be involved in multiple events, e.g., a delivery event and a payment event. An object type can have any number of object attributes, and these attributes should have values. An attribute value of an object can differ across different timestamps, which supports updating a single object attribute instead of updating the complete object. For example, the location attribute of object O_1 can be updated at different timestamps. Different objects may have relationships, e.g., item O_2 is in package O_1. The OCED model resolves the convergence and divergence problems common in convention event data [4]. The

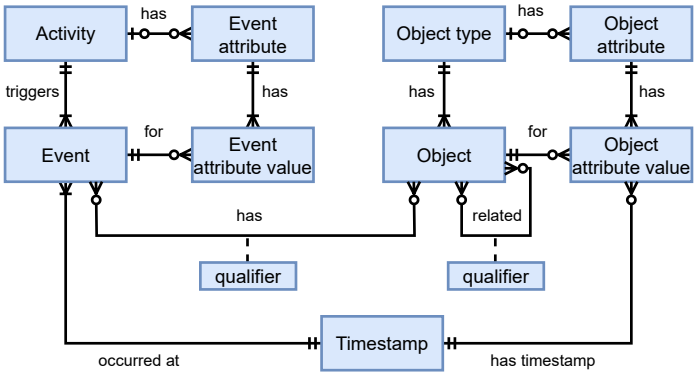

Fig. 2: Object-centric event data model [6]

convergence problem happens when events that involve more than one object are replicated. For instance, when O_1 and O_2 are delivered together, two events are often used to describe this delivery activity in the conventional event data. The divergence problem happens when events that relate to the same object are scattered across different cases. OCED can solve this issue by constructing a case of events for each object. OCED can be used, for instance, in graph embeddings to capture information about the structure of object relationships for subsequent process mining analysis [8].

3.3 User Interaction Event Data

Abb and Rehse [7] presented a User Interaction Event Data (UIED) model for storing low-level activities performed by users during the execution of tasks using software systems. Figure 3 shows the UIED model. Each event in the UIED describes a single interaction between a user and an interface. The activity entity combines the action and the target part of the UIED. It corresponds to the event entity in the conventional event data model. An activity combines a single user and a single target object. The action entity combines the user entity and the activity entity and stores the behavior the user performs using the input devices, such as *click on mouse*. A user can execute any number of actions, while an action can only be performed by exactly one user. The target object entity records the information about the interface object over which the action is executed. The tree-shaped UI component hierarchy is introduced to capture different levels of knowledge about the target object. Each target object refers to such UI hierarchy components. The lowest-level elements of the interface are stored in the UI element entity, such as buttons, text boxes, and their current states. Lowest-level elements can be combined into UI groups based on their functionality, such as the login task. The relationship between UI groups is also modeled in UIED, which is useful for mining the interactions between high-level functions. The UI application and the UI system entities can store higher-level interface information, e.g., the process instance and the system-level action, respectively. Although one can apply OCED to user interaction data and store the information about users and UI objects as objects, UIED has two benefits.

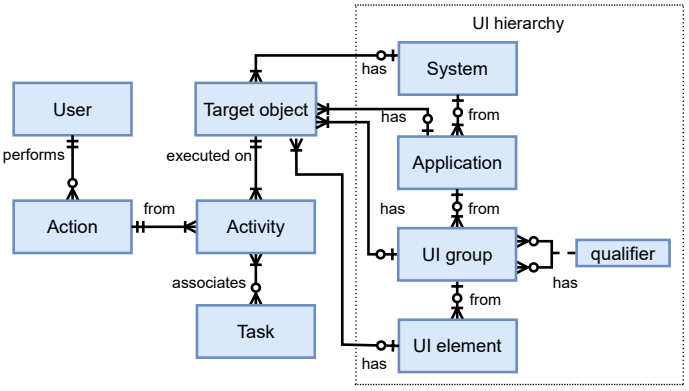

Fig. 3: User interaction event data model [7]

Firstly, for different events, UIED can store dynamic object attributes attached to these events in the entities from the UI hierarchy component. Secondly, object attributes can be shared in different object types based on their UI hierarchy components. However, storing the same target object with different attributes can lead to redundancy.

4 Agent System Event Data

This section presents our Agent System Event Data (ASED) model (Section 4.1), discusses the aspects that characterize ASED as a dimensional model (Section 4.2), explains the relation of ASED to other event data models (Section 4.3), and shows that publicly available event data can already be represented using ASED (Section 4.4).

4.1 Entities and Relationships

Figure 4a shows the ASED model captured using the ER notation. The entities of the model can be split into three categories: conventional process mining entities (highlighted with blue background in the figure), process entity (green), and MAS entities (red). Next, we discuss entities from each of these categories in detail, emphasizing the relevant relationships within and across the categories.

Conventional Entities. Four entities in the ASED model are borrowed from the conventional model. These are the *Event*, *Case*, *Timestamp*, and *Activity* entities. Similar to the conventional process mining, we assume that each event has exactly one process *Case* it refers to and is triggered by exactly one *Activity* at exactly one *Timestamp*. In turn, an *Activity*, *Case*, and *Timestamp* can refer to multiple events. The meaning of these entities is the same as in the conventional process mining; cf. Section 3.1.

Similar to the *Event* and *Case* entities in the conventional data model, every entity of the ASED model can have multiple and arbitrary attribute values, implemented as the *AttributeValue* entity shown in Fig. 4b. This associative entity implements a many-to-many relationship between *Entity* and *Attribute*, with a particular value stored as

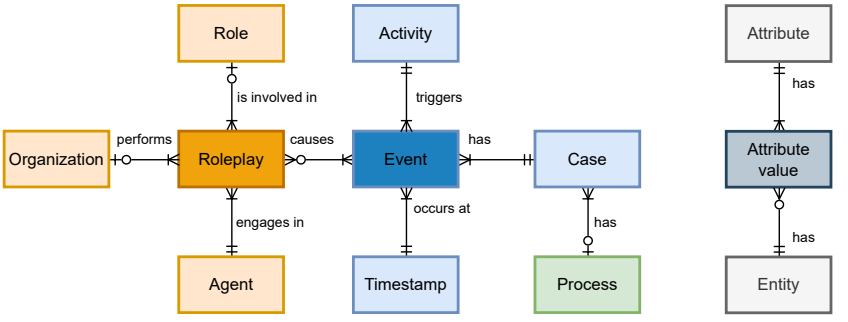

Fig. 4: (a) The Agent System Event Data model and (b) tracking of entity attributes

an attribute of this association. In a concrete implementation of the ASED model, this pattern of attribute values can be replicated for any group of entities in the model. Specifically, every such replica should create copies of *Attribute* and *AttributeValue* entities, specific to the selected group of ASED entities, and link all the entities in the group to the *AttributeValue* entity replica via the one-to-many relationship.

Process Entity. A *Process* is a collection of activities that, when performed, leads to an objective of an organization, for instance, an insurance claim handling process, an implementation of a clinical guideline, or a process of issuing a birth certificate. A *Case* refers to an instance of a process. For example, an insurer usually handles multiple instances of a health insurance process, one per claim, with concrete activities executed to settle the claim. In conventional process mining, it is assumed that the same process triggers all events in an event log. An agent in an organization is often simultaneously involved in multiple instances of different processes. For instance, a financial officer can be simultaneously involved in multiple instances of risk management and financial reporting processes. To understand the essence of the agent's performance, including context switches and task prioritization and transition planning, one needs to expand the scope of event data to the overall landscape of processes in an organization. Thus, in the ASED model, a *Process* must have at least one *Case*, whereas, to ensure compatibility with conventional event data, it is optional for a *Case* to refer to one *Process*.

MAS Entities. An *Agent* is a person or machine that can perform activities to produce specified effects. In an organization, examples of agents include employees, manufacturing equipment, and AI agents performing automated decision-making. A *Role* defines duties and expectations associated with a particular job in an organization, such as specific activities that an agent in this role is responsible for. An agent can relate to multiple roles. For example, an academic can simultaneously relate to the roles of a lecturer to teach classes and a researcher to produce new knowledge. An *Organization* is a group of agents with a particular purpose, for example, a government department, a business, or a healthcare institution. One can also interpret a group of agents as an organizational unit, a distinct team or a division within an organization. Organization units can be structured according to their functional or divisional aspects within organizations. Tracking such structural relationships between organization units is beyond

the scope of the ASED model. A *Roleplay* is then an agent from a specific organization performing an activity to fulfill their duties and responsibilities in a specific role.

Roleplay is an associative entity that captures the many-to-many relationship between agents, roles, and organizations. As already explained, an agent can play multiple roles in an organization, and an organization can involve multiple agents. Similarly, multiple agents can play the same role. Also, the same agent and the same role can be involved in different organizations. We require that a roleplay refers to an agent, whereas it is optional for a roleplay to refer to a role or an organization. In the latter case, we assume that the agent plays some default role in a default organization. The cardinality of the relationship between *Roleplay* and *Event* is many-to-many. Indeed, an agent from an organizatoin playing a role can do that multiple times to cause many events. For instance, an academic from a research group writing a research paper wearing their "scientist hat" can trigger many events, including paper submission, paper review, and camera-ready version submission. At the same time, an event, in general, can be caused by several roleplays. For example, a supervisory meeting event for a research fellow may refer to the fellow, their principal and co-supervisors, collaborators, and research partners, each being in a corresponding roleplay for their organization with respect to this event. To ensure compatibility with the conventional event data, it is optional for an event to refer to a roleplay. In contrast, the existence of a roleplay instance must be justified by relating it to an event.

4.2 Dimensions

Dimensional modeling is a data modeling technique used in data warehousing to organize and structure data for efficient querying and analysis [22, 23]. At the core of dimensional modeling is the concept of a star schema, which consists of a central fact table surrounded by dimension tables. The fact table contains quantitative measures or metrics (facts) related to a business process or event, while dimension tables provide descriptive attributes for analyzing the facts. Dimension tables typically represent entities such as time, location, and objects that characterize the facts in the fact table, and they serve as entry points for querying and filtering data in the fact table. The simple and intuitive structure of a star schema facilitates fast and easy query performance, as well as simplified data retrieval and analysis.

In a star schema, the fact table serves as the focal point for analyzing information and trends, while dimensional tables provide context for interpreting the facts. This model facilitates multidimensional analysis of the data, supporting complex reporting requirements. Additionally, the denormalized structure of a star schema minimizes join operations in queries, leading to improved query performance and reduced database complexity. Overall, dimensional modeling and star schemas provide a flexible and scalable framework for organizing and analyzing data, making it a popular choice for decision support and business intelligence applications.

The ASED model adheres to the structure of a star schema, or a dimensional model, commonly encountered in data warehouses. Within this schema, the *Event* entity and the *Roleplay* entity can be seen as two fact tables, while *Activity*, *Case*, *Timestamp*, *Agent*, *Role*, and *Organization* are dimensional tables that characterize the facts. In addition, the *Process* entity refines the *Case* entity to define a normalized dimensional

hierarchy [23]. It is worth noting that the *Roleplay* entity and the *Event* entity are in a many-to-many relationship, deviating from the typical structure of a star schema in which a fact table is related to dimensional tables via one-to-many relationships. Such a many-to-many relationship can be implemented using an auxiliary table *RoleplayToEvent* in which the related identifier of roleplays and events are matched. Also, note that this deviation from a conventional dimensional design can be addressed through various means. For example, the relationship between *Event* and *Roleplay* can be constrained to a many-to-one relationship by tracking only dominant roleplays for each event [23]. Alternatively, a many-to-many relationship can be interpreted as multiple one-to-one relationships, assuming each event is associated with several roleplays, each corresponding to a distinct role.

We recommend dimensional modeling as a preferred approach for capturing event data in process mining due to its ability to efficiently organize and analyze complex relationships between entities that identify events. By organizing event data into fact tables and dimension tables providing descriptive attributes, dimensional modeling facilitates the exploration and analysis of events from multiple perspectives. This approach allows process mining practitioners to quickly identify patterns, trends, and correlations within event data, enabling insights into process behavior, performance, and compliance. The denormalized structure of star schemas supports faster query performance and streamlined data retrieval [24]. Furthermore, the multidimensional nature of dimensional modeling supports advanced analytical techniques, such as OLAP (Online Analytical Processing), enabling interactive and dynamic exploration of event data across various dimensions. Overall, dimensional modeling offers a robust framework for capturing, organizing, and analyzing event data in process mining, empowering organizations to extract valuable insights and optimize their business processes effectively.

4.3 Compatibility

ASED can be applied in different process mining paradigms, such as Agent System Mining, Robotic Process Mining (or Task Mining), and Object-Centric Process Mining, see Section 2 for the discussion of these process mining types. In this section, we discuss the relation between ASED and event data models used in these process mining types.

Agent System Mining. Agent System Mining (ASM) requires a collection of event data, each characterized by a case identifier, a timestamp, an activity, and an agent. In classical event data, the case identifier and activity that triggered an event are stored within the *Case* entity and the *Event* entity, respectively, and the agent information is stored within the event attributes. In ASED, all agent attributes are stored in the *Agent* entity, whereas the other three types of information are stored in the same entities as in classical event data. Existing algorithms for discovering agent-based models from event data in ASM require that every event is triggered by exactly one agent [29]. These algorithms can be extended to benefit from the ability of ASED to express the fact that an event can be triggered by several agents playing different roles in different organizations. In addition, ASED, for the first time, argues that event data may encompass data stemming from multiple processes, a phenomenon that ASM can benefit from when analyzing agents' performance across different functions [28]. Future work in ASM will

explore the benefits of ASED and the boundaries of what models can be inferred based on the available concepts, and which analysis these concepts can support.

Task Mining. ASED can be applied to support user interaction mining techniques. One can interpret the *Agent* and *Role* entities in ASED as the *User* and *Action* entities in UIED, respectively, while the *Event* entity can be interpreted as the *Activity* entity in UIED. In ASED, multiple agents (users) can be involved in the same event, e.g., an online meeting. The information related to these user actions is stored in the Roleplay entity. This many-to-many relationship is useful when doing task mining on collaboration platforms that track interactions between users. In terms of dimensional modeling, the *Event* entity in ASED is a fact table characterized by the dimensions referred to by the keys of the *Roleplay*, *Activity*, and *Timestamp* entities. Hierarchical dimensional tables can also be used to characterize the *Target object* entity to store attributes from the UI hierarchy component. Such dimensional tables can support efficient querying and are suitable for mining the relationships between different UI levels. Furthermore, the *Task* entity in UEID can be interpreted as the *Process* entity in ASED, while the *Target object* entity in UIED can be accepted as the *Activity* entity in ASED. Consequently, one can combine the features of ASED and UIED by linking the UI hierarchy concepts in UIED to the *Activity* entity in ASED, obtaining the benefits of both designs.

Object-Centric Process Mining. Object-Centric Event Data (OCED) can be integrated with ASED by relating the *Event* entity in ASED with the *Object* entity in OCED. The relationship between the ASED *Event* entity and the OCED *Object* entity should also be borrowed from the OCED model; that is, it is a many-to-many relationship. The other entities and relationships from both ASED and OCED should then be included in the integrated event data model without further modifications. In such an integrated data model, one can capture relationships between agents, events, and objects, benefiting from both original designs. Consequently, according to the integrated model, an event can be triggered by any number of agents playing different roles in their organizations. At the same time, an event in such an integrated world can be related to any number of objects, providing scopes for identifying process cases as various combinations of objects. This ability to identify process cases as combinations of objects involved in the occurrence of events constitutes the main benefit of OCED over the conventional event data model [5]. Such an integrated data mode supports the querying of the information between agents and objects. In addition, it can be used to study resource allocation across various processes and cases. Finally, such an integrated model supports tracking of the relationships between objects and processes and between objects and organizations. This is particularly useful for relating business objects to relevant departments and processes within organizations.

4.4 Feasibility

As ASED is a new data model, there are no publicly available event datasets captured in ASED. In this section, we study the feasibility of obtaining an event dataset that adheres to the ASED requirements by converting a publicly available conventional event log to ASED. Specifically, we convert the BPIC 2013 event log[4] into the ASED format. The

[4] https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee

BPIC 2013 event log is an extract of the Volvo IT Incident and Problem Management event log, which stores information about agents, resources, organizations, and processes as event attributes. We implemented the ASED model as a relational database.[5] The attributes of the various ASED entities were obtained based on the available attributes of the *Event* entity from the BPIC 2013 event log. The implementation of the conversion is publicly available.[6] We also make publicly available the result of the conversion of the BPIC 2013 event log into the ASED dataset. To facilitate accessibility of the results, we store the tables of the derived schema in the CSV format[7], and keep the timestamp values in the event fact table. In the converted ASED event log, each event is performed by a single roleplay. However, in general, it is possible to map conventional events to ASED events performed by several roleplays. For example, such a situation may arise when the attributes of some specific events contain information about more than one agent, their roles, and organizations. Alternatively, one can decide to cluster conventional events into high-level ASED events and model them as such caused by all the roleplays involved in the corresponding low-level events. However, such a clustering step may require additional information besides the conventional event log, e.g., configuration of the clustering technique or manual definitions of high-level events.

5 Process Compliance

Next, we validate the ability to express process compliance rules that relate to the management of agents, roles, and organizations in processes over the ASED data model. To identify relevant compliance rules, we performed a literature search. We executed the query "*(process compliance) AND (role OR resource OR agent OR object)*" on Web of Science to search for relevant papers. On May 17, 2024, this query returned 58 papers. After screening the titles, abstracts, and the full content of the papers, we identified 8 papers that discuss process compliance rules of interest. We examined these papers and extracted *all* relevant compliance rules from them, resulting in a total of 40 identified rules. Each of these compliance rules was evaluated to determine if it can be implemented over all event data models discussed in Section 3 and ASED. The conventional event data and OCED cannot express any of these 40 compliance rules, while UIED can express some of the rules that address the relationship between agents and activities. The summary of the rules supported by ASED and UIED is given in Table 1.

The identified compliance rules address relationships between multiple attributes, such as role, case, agent, and time. Most of these rules can be implemented over the ASED model, as is documented in the last column of Table 1. We implemented all compliance rules supported by ASED.[8] In addition, we converted BPIC 2013, a publicly available industrial event log, to ASED.[7] We evaluated the implemented rules over the obtained ASED model to validate the claims of ASED support in Table 1.

For example, one can implement the rule "A_1 *(an agent) should not play different roles in the same case*" (Rule 18 in Table 1) by checking, for each case, if A_1 is in-

[5] https://doi.org/10.26188/26868532.v1

[6] https://doi.org/10.26188/26868586.v1

[7] https://doi.org/10.26188/26868592.v1

[8] https://doi.org/10.26188/26868442.v2

Table 1: Process compliance rules, where T_1, T_2, and T_3 refer to tasks/activities, A_1 and A_2 refer to agents, R_1 and R_2 refer to roles, O_1 and O_2 refer to organizations (organization units), P_1 and P_2 refer to processes, and M refers to a time period; "✓" – rule supported; "×" – rule not supported; "AS" – Agent System Event Data; "UI" – User Interaction Event Data

No.	Compliance rules	Ref.	UI	AS
1	T_1 must be done by A_1	[11]	✓	✓
2	A_1 is responsible for T_1	[11]	×	×
3	T_1 must be done by R_1	[11, 13, 26]	×	✓
4	T_1 must be done by O_1	[11]	×	✓
5	T_1 must be done by an agent in list $[A_1, A_2, \ldots]$	[11]	✓	✓
6	An agent in list $[A_1, A_2, \ldots]$ is responsible for T_1	[11]	×	×
7	T_1 must be done by a role in list $[R_1, R_2, \ldots]$	[11, 18]	×	✓
8	T_1 must be done by an organization in list $[O_1, O_2, \ldots]$	[11]	×	✓
9	T_1, T_2 should not be done by the same agent in one case	[13]	×	✓
10	T_1, T_2 should not be done by the same role in one case	[13]	×	✓
11	T_1, T_2 must be done by the same agent in one case	[13]	×	✓
12	T_1, T_2 must be done by the same role but different agents in one case	[13]	×	✓
13	R_1, R_2 must be involved together in T_1	[13]	×	✓
14	A_1, A_2 must be involved together in T_1	[13]	×	✓
15	A_1 should not play only role R_1 in the whole process	[12]	×	✓
16	The number of roles A_1 plays in the whole process is limited	[12]	×	✓
17	The number of cases A_1 participates in is limited	[12]	×	✓
18	A_1 should not play different roles in the same case	[12]	×	✓
19	If A_1 does T_1 in R_1, then A_1 should not do T_2 in R_2 in the same case	[12]	×	✓
20	If A_1 does T_1, then A_1 must perform T_2 in same case	[17]	×	✓
21	If O_1 does T_1, then this T_1 must be done by A_1 in O_1	[17]	×	✓
22	If A_1 in O_1 does T_1, then O_2 must perform T_2 earlier in same case	[17]	×	✓
23	If O_1 does T_1, then O_2 must perform T_2 later in same case	[17]	×	✓
24	If T_1, T_2 are directly followed, then they cannot done by a same agent	[17]	×	✓
25	R_1 is always responsible for R_2	[27]	×	×
26	Number of activities done by A_1 in the whole process is limited	[18]	×	✓
27	Number of activities done by R_1 in the whole process is limited	[18]	×	✓
28	Two tasks from two cases of different processes should not be done by the same role	[18]	×	✓
29	At least one activity in a case must be done by roles in the middle level	[18]	×	×
30	R_1 can perform a maximum of N activities in any case of P_1	[18]	×	✓
31	T_1, T_2 should be done by different organizations	[25]	×	✓
32	If A_1 does T_1 and not T_2 in period M, then A_1 must do T_3 later	[25]	✓	✓
33	If A_1 does T_1 and not T_2 in period M, then R_1 must do T_3 later	[25]	×	✓
34	T_1 must be done by an agent with R_1 in O_1	[26]	×	✓
35	If T_1 occurs before T_2 in a case, they must be done by the same agent	[26]	✓	✓
36	If T_1 occurs before T_2 in a case, they must be done by agents from O_1	[26]	×	✓
37	If T_1 is done by A_1 in R_1, T_2 should be done by a lower-level role before	[26]	×	×
38	If T_1 and T_2 are done by the same organization and T_3 occurs, T_3 should be done by R_1	[26]	×	✓
39	If A_1 does T_1, T_2 sequentially, then A_1 must in O_1	[26]	×	✓
40	If T_1, T_2 occurs before T_3, then T_1, T_2 must be done by agents in O_1	[26]	×	✓

Table 2: (a) Instances of the *Event* entity, (b) instances of the *Roleplay* entity, and (c) relationships between the *Event* and *Role* entities

(a)

ID	Case	Activity	Timestamp
E_1	C_1	T_1	*2010-03-31 14:59*
E_2	C_1	T_1	*2010-03-31 15:00*
E_3	C_1	T_2	*2010-03-31 15:45*
E_4	C_1	T_1	*2010-04-06 14:44*
E_5	C_1	T_2	*2010-04-06 14:45*

(b)

ID	Agent	Role	Organization
RP_1	A_1	R_1	O_1
RP_2	A_1	R_2	O_1
RP_3	A_2	R_2	O_1
RP_4	A_2	R_1	O_1
RP_5	A_3	R_1	O_1

(c)

Event	Roleplay
E_1	RP_1
E_2	RP_1
E_3	RP_2
E_4	RP_3
E_5	RP_4

volved only in one role associated with all events that relate to that case. This logic over the ASED model is captured in Algorithm 1. Table 2a and Table 2b lists instances of the *Event* and *Role* entities, respectively, while Table 2c implements the *EventToRoleplay* relationship by relating events to roleplays that were involved in triggering them. Consider that Table 2a stores all events from case C_1. Table 2c suggests that four roleplays were involved in this case. These roleplays can be used to extract roles and agents that define them; see Table 2b. Assume that Algorithm 1 is invoked for the input of agent identifier A_1 and entity instances and relationships in Tables 2a to 2c. As there is only one case (C_1), the `for` loop at Line 2 of the algorithm will execute once. In this only iteration, the *eids* variable will be assigned set $\{E_1,E_2,E_3,E_4,E_5\}$; this is the set of all events in the currently analyzed case (cf. Line 3). Then, at Line 4, all roleplays relevant to the events in *eids* are extracted. Hence, it holds that *rids* is equal to $\{RP_1,RP_2,RP_3,RP_4\}$. At Line 5, the algorithm computes all roles the requested agent (A_1) played in roleplays from *rids* and stores them in *roles*. If the number of played roles exceeds one ($|roles| > 1$), the case is added to the *res* set, which was initialized as the empty set at Line 1. If the returned set *res* is empty, there are no cases in which the agent plays more than one role, and thus, the rule is satisfied; otherwise, the rule is violated. As *res* contains C_1, compliance rule 18 is violated on this example input.

The ASED model does not support five of the surveyed compliance rules (cf. Table 1). Rules 2 and 6 refer to the responsibilities of agents when performing activities. For example, Rule 2 "A_1 *is responsible for* T_1" requests to check that agent A_1 always acts in the highest role when performing activity T_1. This check can be performed in two steps. Firstly, one needs to confirm that A_1 is involved in the execution of T_1. Secondly, when a group of agents is involved in the execution of T_1, one needs to confirm that A_1 has the highest-level role in this agent group [11]. ASED supports checking the first step, while the second step cannot be checked because ASED does not explicitly model the hierarchy of roles. Therefore, Rules 2 and 6 cannot be checked over ASED. Rule 25 requests that "R_1 *is always responsible for* R_2," meaning that if R_2 is involved in a case, then R_1 must also be involved in the same case, and the role level of R_1 must be higher than R_2 [18]. Again, we cannot check this rule without knowing the role levels. The knowledge of role hierarchy is also necessary when checking Rules 29 and 37. Therefore, to support checking these five rules, one should model role levels explicitly or infer them from available data.

It is significantly more challenging to check the compliance rules from Table 1 over other event data models. In the conventional model, it is impossible to check the rules

Algorithm 1: Compliance rule 18 from Table 1 over ASED

Input: An agent identifier *aid*, instances of the *Event* and *Roleplay* entities, and an *EventToRoleplay* relationship between the *Event* and *Roleplay* entities.
Output: The set of all case identifiers in which agent A_1 plays more than one role.

$res \leftarrow \emptyset$;
for $c \in Case$ **do**
  $eids \leftarrow \{e.ID \mid e \in Event \wedge e.Case = c.ID\}$;
  $rids \leftarrow \{r.Roleplay \mid e2r \in EventToRoleplay \wedge e2r.Event \in eids\}$;
  $roles \leftarrow \{role \mid r \in Roleplay \wedge \in r.ID \in rids \wedge r.Agent = aid\}$;
  if $|roles| > 1$ **then** $res \leftarrow res \cup \{c.ID\}$;
return *res*

by relying only on event attributes, as the semantics of such attributes are unspecified. For instance, it is not possible to match agents and roles if an event has two attributes storing information about agents and two attributes storing information about roles unless some additional knowledge about the mapping is available. Specifically, the event "*hold a meeting*" may involve many participants in different roles, and it may be necessary to relate participants to corresponding roles. OCED does not model agents, roles, roleplays, or organizations explicitly. However, the generic nature of OCED, implemented via objects and their qualified relationships, allows for the encompassing of any data schema, including ASED. This approach, however, requires external schema knowledge that is not initially available in OCED. The original OCED model, hence, cannot support any compliance rules listed in Table 1. UIED can describe hierarchies of roles and store agent information in the target object entity. However, it is not possible to express that a specific agent can play different roles. These aspects of UIED make it suitable for implementing only four compliance rules out of those listed in Table 1.

As all compliance rules currently not supported by ASED relate to responsibilities of agents [10], the extension of ASED toward modeling of responsibilities of agents or inference of responsibilities based on ASED is future work.

6 Conclusion

This paper presents Agent System Event Data (ASED), a model aimed at conceptualizing the event data used in Agent System Mining. We provide a detailed exposition of ASED and conduct a comprehensive review of various event data models in process mining, elucidating the relationship between ASED and the existing models. Furthermore, we subject ASED to a rigorous validation by demonstrating how ASED can be used to verify a diverse array of process compliance rules pertaining to roles, agents, and resources. Through this validation, we confirm that ASED significantly expands the coverage of compliance rules supported by existing event data models.

Several limitations of this work should be acknowledged. First, one can conduct a systematic literature review to expand the list of identified relevant compliance rules used to validate the ASED model. Second, ASED currently lacks the capability to model responsibilities of agents and superior-subordinate relationships between agents.

Lastly, the discussion of ASED's feasibility does not account for the practicality of directly recording the data generated by existing information systems. Future work will aim to address these limitations. Lastly, we plan to explore the ability of ASED to facilitate the discovery of agent-based models from event data.

Acknowledgment. This work was in part supported by the Australian Research Council project DP220101516.

References

[1] IEEE Standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2023 (Revision of IEEE Std 1849-2016) (2023)
[2] van der Aalst, W.M.P.: Process mining: Overview and opportunities. ACM Trans. Manag. Inf. Syst. **3**(2), 7:1–7:17 (2012)
[3] van der Aalst, W.M.P.: Process Mining—Data Science in Action. 2nd edn. (2016)
[4] van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: SEFM, Lecture Notes in Computer Science, vol. 11724, pp. 3–25, Springer (2019)
[5] van der Aalst, W.M.P.: Object-centric process mining: An introduction. In: ICTAC Summmer School, Lecture Notes in Computer Science, vol. 13490, pp. 73–105, Springer (2021)
[6] van der Aalst, W.M.P.: Object-centric process mining: Unraveling the fabric of real processes. Mathematics **11**(12), 2691 (2023)
[7] Abb, L., Rehse, J.: A reference data model for process-related user interaction logs. In: BPM, Lecture Notes in Computer Science, vol. 13420, pp. 57–74, Springer (2022)
[8] Adams, J.N., Park, G., van der Aalst, W.M.P.: Preserving complex object-centric graph structures to improve machine learning tasks in process mining. Eng. Appl. Artif. Intell. **125**, 106764 (2023)
[9] Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Sci. Comput. Program. **78**(6), 747–761 (2013)
[10] Cabanillas, C., Resinas, M., Cortés, A.R.: Automated resource assignment in BPMN models using RACI matrices. In: OTM Conferences (1), Lecture Notes in Computer Science, vol. 7565, pp. 56–73, Springer (2012)
[11] Cabanillas, C., Resinas, M., Cortés, A.R.: Introducing a mashup-based approach for design-time compliance checking in business processes. In: CAiSE Workshops, Lecture Notes in Business Information Processing, vol. 112, pp. 337–350, Springer (2012)
[12] Dalpiaz, F., Cardoso, E., Canobbio, G., Giorgini, P., Mylopoulos, J.: Social specifications of business processes with Azzurra. In: RCIS, pp. 7–18, IEEE (2015)
[13] Elgammal, A., Türetken, O., van den Heuvel, W., Papazoglou, M.P.: Formalizing and appling compliance patterns for business process compliance. Softw. Syst. Model. **15**(1), 119–146 (2016)
[14] Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: ICMAS, pp. 128–135, IEEE Computer Society (1998)
[15] Hobeck, R., Weber, I.: Towards object-centric process mining for blockchain applications. In: BPM Blockchain and RPA Forum, Lecture Notes in Business Information Processing, vol. 491, pp. 51–65, Springer (2023)
[16] Ito, S., Vymetal, D., Sperka, R., Halaska, M.: Process mining of a multi-agent business simulator. Comput. Math. Organ. Theory **24**(4), 500–531 (2018)
[17] Knuplesch, D., Reichert, M.: A visual language for modeling multiple perspectives of business process compliance rules. Softw. Syst. Model. **16**(3), 715–736 (2017)

[18] Kumar, A., Liu, R.: A rule-based framework using role patterns for business process compliance. In: RuleML, Lecture Notes in Computer Science, vol. 5321, pp. 58–72, Springer (2008)
[19] Lassen, K.B., van Dongen, B.F.: Translating message sequence charts to other process languages using process mining. Trans. Petri Nets Other Model. Concurr. **1**, 71–85 (2008)
[20] Leno, V., Polyvyanyy, A., Dumas, M., Rosa, M.L., Maggi, F.M.: Robotic process mining: Vision and challenges. Bus. Inf. Syst. Eng. **63**(3), 301–314 (2021)
[21] Li, G., de Carvalho, R.M.: Process mining in social media: Applying object-centric behavioral constraint models. IEEE Access **7**, 84360–84373 (2019)
[22] Moody, D.L., Kortink, M.A.: From ER models to dimensional models: Bridging the gap between OLTP and OLAP design, Part 1. Journal of Business Intelligence **8**, 7–24 (2003)
[23] Moody, D.L., Kortink, M.A.: From ER models to dimensional models Part II: Advanced design issues. Journal of Business Intelligence **8**, 20–29 (2003)
[24] Polyvyanyy, A.: Process Querying Methods. Springer, Cham (2022)
[25] Sadiq, S.W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: BPM, Lecture Notes in Computer Science, vol. 4714, pp. 149–164, Springer (2007)
[26] Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: BMMDS/EMMSAD, Lecture Notes in Business Information Processing, vol. 175, pp. 48–63, Springer (2014)
[27] Soto, M., Münch, J.: Focused identification of process model changes. In: ICSP, Lecture Notes in Computer Science, vol. 4470, pp. 182–194, Springer (2007)
[28] Tour, A., Polyvyanyy, A., Kalenkova, A.A.: Agent system mining: Vision, benefits, and challenges. IEEE Access **9**, 99480–99494 (2021)
[29] Tour, A., Polyvyanyy, A., Kalenkova, A.A., Senderovich, A.: Agent miner: An algorithm for discovering agent systems from event data. In: BPM, Lecture Notes in Computer Science, vol. 14159, pp. 284–302, Springer (2023)