Applying Organizational Mining to Discover Agent Systems from Event Data

Qingtan Shen[1], Artem Polyvyanyy[1], Nir Lipovetzky[1], and Timotheus Kampik[2,3]

[1]The University of Melbourne, Victoria 3010, Australia
[2]SAP, Germany
[3]Umeå University, Sweden

Tuesday 23rd December, 2025

Abstract

Agent system mining is a recently introduced type of process mining that takes a bottom-up approach to the data-driven analysis of socio-technical systems that execute business processes in organizations. Instead of the top-down approach used in conventional process mining that studies a system in terms of its global state evolution, agent system mining analyzes the system as if it is composed of autonomous agents, each with its local state and behavior, interacting with other agents and the environment to contribute to the emerging global behavior of the business process. Recently, Agent Miner, the first algorithm for discovering agent systems from event data generated by process-aware information systems, has been proposed. The quality of the agent systems discovered by this algorithm depends on the quality of the agent types (or agents), which are identified from the available information about agent instances in the data. In this paper, we study the suitability and benefits of using methods from the organizational mining subarea of process mining for identifying agent types. The experiments we conduct over real-world datasets confirm the usefulness of such methods for discovering simple, modular, and accurate agent systems. These conclusions are grounded in quality metrics such as the size of discovered models (simplicity), Louvain modularity and the Gini coefficient (modularity), and precision and recall (accuracy). The results confirm the benefits of using organizational mining for identifying agent types when discovering agent systems from event data, leading to the construction of models of superior quality in precision, recall, and simplicity compared to models constructed by state-of-the-art conventional process discovery algorithms.

Keywords: Process mining, agent system mining, agent system discovery

1 Introduction

Process mining analyzes event data generated by organizational processes and recorded by information systems to improve future processes [43]. Depending on the perspective taken on system behavior, process mining techniques can be divided into

three main categories: activity-oriented, product-oriented, and agent-oriented techniques [18]. Agent system mining is a type of agent-oriented process mining that follows a bottom-up approach for process analysis [39]. In this approach, the systems that generate the process data are assumed to be composed of autonomous agents interacting with each other and the environment to achieve their own and the organizations' goals; that is, they are agent-based systems. The processes that are the focus of conventional process mining analysis are then accepted as the result of the emerging behavior of the agent-based systems. Agent system mining involves expanding the analysis focus of process mining beyond the processes themselves to include the participants' local processes and their interactions. The goal is to gain a comprehensive understanding of the components and principles governing the dynamics of the system to improve its future operations and process outcomes. Conceptually, one may argue that agent system mining is of interest to researchers and practitioners for two reasons: *i)* it bridges process mining and artificial intelligence research, considering that the latter is commonly defined as "the study of agents that receive percepts from the environment and perform actions" [29, p. viii]; *ii)* it reconciles analyses of micro-level behaviors of agents with macro-level process dynamics. Thus, it acknowledges that the notion of a *process* is, from the agent-oriented perspective, an artificial abstraction used to govern and analyze socio-technical systems that allow for substantial autonomy of (human or software) subsystems.

Agent Miner [40] is an algorithm for discovering models of agents and their interactions from event data generated by agent-based systems. In process mining, event data is usually given as an *event log*, a collection of recorded events, each characterized by a case identifier, timestamp, and activity. Events that refer to the same *case* stem from the same process instance, for example, a process of handling one patient at a hospital. The *timestamp* indicates the time at which the event has occurred. Finally, the *activity* attribute specifies the activity that triggered the event. Agent Miner requires that, in addition to the three standard attributes, each event specifies the *agent*, or *agent instance*, which is an entity, e.g., a person or a system component, that executed the activity that triggered the event. While each event originates from a single agent instance, the overall behaviour of that agent can be captured as all ordered sequences of the activities it performs over time. This agent-related information is available in most real-world event logs, though it may be recorded under different event attributes, such as *resource* or *resource ID*. All ten real-world event logs used in our experiments include this information, making them suitable for our analysis.

Agent type identification is a key step in the Agent Miner algorithm. An event log can contain information on multiple agent instances executing various activities. These agent instances often perform similar tasks to support the high volume of process cases. For example, it is common for multiple employees to handle customers' queries while other employees focus on issue resolution. If one can identify these different types of agents effectively, that information can be used to create accurate agent models and their composition into a system that is faithful to the corresponding real system. Consequently, the problem of *agent type identification* studies ways to use event logs to infer groups of agent instances that perform similar process activities that lead to the construction of agent-based systems that accurately represent the system that generated the data. Agent-based models can also be used in process simulation to

generate synthetic event data that reflects the behavior of the original system [36]. To this end, the use of multi-agent system (MAS) models generated from agent system mining for simulation has been proposed in the MAS modeling and simulation lifecycle [39]. Accordingly, Petri net-based MAS models generated by Agent Miner can be directly applied to agent-based simulation tasks, further demonstrating its practical value [19].

In this work, we study the benefits of using organizational mining techniques for solving the problem of agent type identification. Organizational mining is a branch of process mining that studies techniques to discover organizational models, roles, and social networks from event data and how these artifacts can be used to improve the processes [5, 27, 35]. The discovery of organizational knowledge supports the understanding of organizational structures, which can be used to improve information flows between the process participants.

We review state-of-the-art approaches to organizational mining as a foundation for developing agent type identification techniques. Based on this review, we identify three approaches that can be applied directly or serve as a basis for designing agent type identification techniques. These three approaches are Comprehensive Workflow Mining (CWM) [27], the Activity Matrix (AMatrix) method [35], and the Business Models Enhancement (BME) approach [5]. The detailed review process and the explanation of how these three approaches were identified are presented in Section 4.2.1. Through extensive experiments over industrial event logs, we conclude that agent type identification techniques grounded in organizational mining methods substantially improve the diversity of models discovered by Agent Miner in terms of accuracy (precision and recall) and simplicity (model size) characteristics. Notably, the discovered agent-based models are often comparable with process models constructed by the state-of-the-art conventional process discovery algorithms in terms of accuracy and simplicity. Moreover, in some situations, they can outperform the discovered process models in both accuracy and simplicity simultaneously.

Specifically, we make the following contributions:

1. We review existing organizational mining methods and identify three methods, CWM, AMatrix, and BME, that can be reused to perform agent type identification.
2. We use existing organizational mining methods to support the inference of agent types from event data. While AMatrix and BME can be used off-the-shelf, we show how to use the ideas from CWM for our specific purpose in the approach we refer to as Dominant Cluster Grouping (DCG).
3. We evaluate the quality of agent-based models discovered by the native approach of Agent Miner and three new agent type identification methods, and confirm the usefulness of the newly proposed organizational mining methods. To this end, we assess traditional quality criteria, i.e., precision, recall, and size, as well as the modularity criterion pertinent to agent models, i.e., Louvain modularity and the Gini coefficient.
4. We demonstrate that discovered models often outperform process models discovered by the state-of-the-art process discovery algorithms in precision, recall, or simplicity and, remarkably, can outperform discovered process models in all these qualities simultaneously.
5. We assess the accuracy of the agent types inferred by the presented agent type iden-

tification techniques using synthetic event logs obtained by simulating MAS models with established ground truth agent types. The findings suggest that the AMatrix approach yields the highest accuracy.

The next section provides background knowledge on process mining and agent system mining. In Section 3, we introduce and motivate the problem of agent type identification through an illustrative example. Section 4 details four agent type identification techniques: the one currently employed by Agent Miner and three new approaches adapted from reviewed organizational mining techniques. Section 5 presents the evaluation results for all the techniques, highlighting those that lead to the discovery of simple and accurate models, and discusses potential threats to the validity of these findings. Finally, Section 6 offers concluding remarks and outlines future directions.

2 Background

This section gives background knowledge on process mining and agent system mining.

2.1 Process Mining

Process mining studies ways to discover, monitor, and enhance real-world processes by extracting knowledge from event logs generated by information systems [42]. Process discovery, a key subarea of process mining, aims to construct process models from event logs automatically. These constructed models can be utilized to visualize, analyze, and optimize real-world processes. Over the past two decades, numerous process discovery techniques have been proposed. Earlier process discovery techniques include α-miner and heuristic miner. The Alpha Miner [45] algorithm constructs Petri nets from event logs, while Heuristic Miner [47] generates heuristic nets comprising causal relations with frequency annotations. Heuristic Miner, and later introduced techniques like Fuzzy Miner [17] and Genetic Miner [11], are effective in handling noise and incompleteness of event logs when constructing process models. Finally, Inductive Miner (IM) [20], one of the state-of-the-art process discovery techniques [43], constructs a block-structured process model, captured as a process tree, that is both correct, in terms of soundness [41], and accurately describes the system that generated the input event log. IM employs a divide-and-conquer strategy to recursively identify model constructs and refine them into a resulting process model, but it relies on a noise threshold to filter infrequent behavior. Different process discovery algorithms generate process models expressed through different syntaxes, but they are often mapped to Petri nets to explain their formal semantics. Figure 1 illustrates an example of a process model, captured as a Petri net, discovered by IM from the BPIC 2014 (incident details)[1] event log. This event log contains records on incidents tracked at Rabobank Group ICT.

The widely used criteria for assessing the quality of process models discovered from an event log are precision, recall, and simplicity [4, 37]. Precision and recall

[1] BPIC 2014 (incident details) is one of many industrial event logs made publicly available by the IEEE Task Force on Process Mining as part of the yearly Business Process Intelligence Challenge (BPIC) contest. It can be downloaded via https://doi.org/10.4121/uuid:3cfa2260-f5c5-44be-afe1-b70d35288d6d.

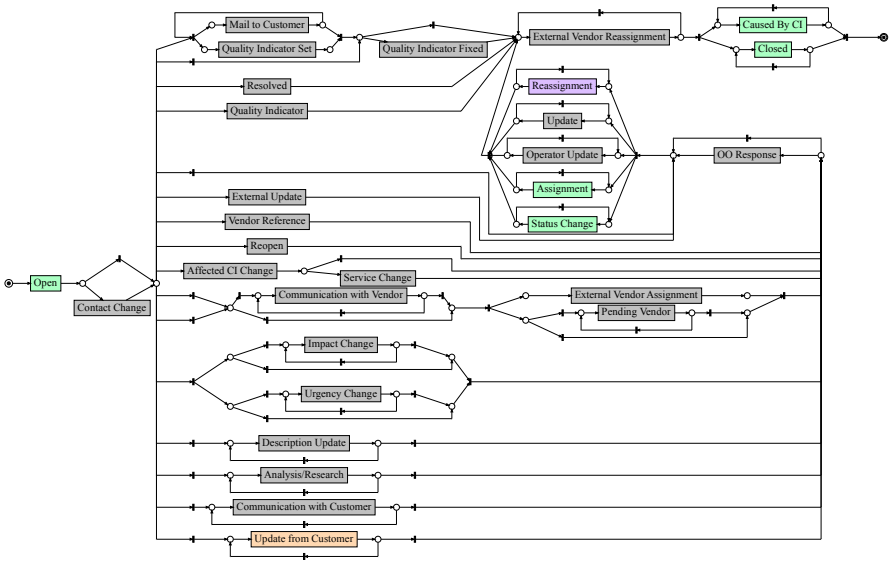


Figure 1: The process model captured as a Petri net discovered by Inductive Miner from the BPIC 2014 (incident details) event log using a noise threshold of 0.26. The model has precision of 0.33, recall of 0.55, and size of 322.

compare a given event log and process model based on traces they describe. Each case identifier in an event log induces a trace as the sequence of activities that relate to all the events that refer to that case identifier, ordered by the timestamps of the events. A trace of a process model is a sequence of activities that can be executed according to the semantics of the model. A model with good precision does not describe many traces not included in the event log, while a model with good recall describes many traces recorded in the event log. Precision and recall measurements usually take values between zero and one, inclusive, with larger values denoting better models. Finally, a model that is easy to understand and analyze, for example, because it avoids redundant activities and intricate control flow constructs, is considered to be simple.

To measure the simplicity of a process model, in this work, we use its size, that is, the number of nodes and edges in the model, with models of smaller sizes considered to be simpler. To quantify precision and recall, we use the Entropia tool [24] to compute entropy-based measures [25]. The entropy-based measures of precision and recall are the only measures reported by Syring et al. [37] that fulfill *all* the properties put forward by the process mining community for measures that assess these quality criteria of discovered models. For example, one of these properties ensures that for two process models and one event log, if the first model describes all the traces in the event log and the traces described by this model are all described by the second model, then the first model should have a precision score with respect to the event log that is at least as high

as the precision score of the second model with respect to this event log.[2] Intuitively, the fact that we use entropy-based precision and recall measures *ensures* that better precision and recall measurements we report correspond to better process models, that is, models that describe the system that generated the event log more faithfully. The Petri net model in Fig. 1 has the entropy-based precision and recall with respect to the event log it was constructed from of 0.33 and 0.55, respectively, and is composed of 322 nodes (places and transitions) and edges.

2.2 Agent System Mining

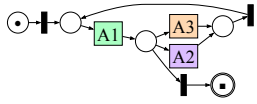
MASs are widely studied in different disciplines, including computer science and civil engineering [13]. Agents in MASs are typically characterized by three core features: sociability, autonomy, and proactivity [13]. Sociability refers to the agent's ability to communicate and interact with other agents, enabling knowledge sharing and information requests that support their goal achievements. Autonomy refers to an agent's ability to make decisions and take corresponding actions independently. Finally, proactivity refers to an agent's ability to pursue goals and perform actions without being triggered by external events.

An MAS consists of multiple agents that interact to achieve shared objectives. MASs are often characterized by self-organization, which refers to the ability of agents to autonomously coordinate and operate based on contextual local interactions, without the need for centralized control [33]. Emergence refers to the formation of coherent behavioral patterns that arise during the process of self-organization in complex systems. In MASs, macro-level emergent phenomena are system-level behaviors that arise from the interactions of agents and cannot be inferred by analyzing individual agents in isolation [33]. MAS design methodologies typically follow either a top-down or a bottom-up approach [9]. The top-down approach begins with specifying the global system state and assuming that each component has knowledge of the overall system state. Conversely, the bottom-up approach starts with specifying requirements and capabilities of individual components, from which global system behavior emerges through interactions among agents and their environment.

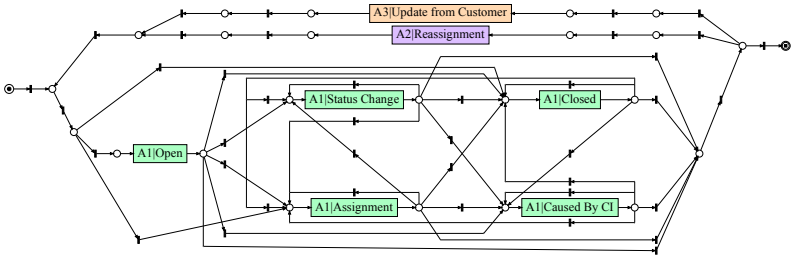
Business processes where agents act autonomously while collaborating to fulfill organizational goals can be effectively modeled as MASs. Agent-Based Modeling (ABM) is a method used to represent organizations as systems of autonomous, interacting agents [22]. Unlike traditional process discovery techniques, which focus on constructing process models, *agent system mining* integrates ABM with process mining to automatically construct MAS models from event data [39]. A recent systematic literature review studies the role of process mining in agent-based modeling and simulation [2], providing further insights for developing agent system mining.

A MAS is often captured using two types of models: agent models and an (agent) interaction model. Agent models describe the behaviors of individual agents, while the interaction model captures patterns of communication and collaboration among agents. Agent system mining provides at least the following three benefits compared to

[2]This property of a precision measure is axiom *A2* presented by Tax et al. [38] and a partial case of property $PrecPro1^+$ proposed by van der Aalst [44].



(a)



(b)

Figure 2: (a) The interaction model and (b) MAS model discovered by Agent Miner from the BPIC 2014 (incident details) event log using the activity frequency filter of 0.1, level of details threshold for the interaction model of 0.9, and the BME agent type identification technique. The MAS model has precision of 0.40, recall of 0.71, and size of 166; both models are captured as Petri nets.

classical process discovery methods [39]. First, it analyzes both the behavior of process participants and their interactions, ensuring that related activities performed by the same agent are grouped within the model (Fig. 2b). Second, as the volume of available event data increases, MAS models that aim to describe this data do not necessarily grow in size [48]. This allows for simple yet accurate representations of complex processes, providing clear scopes of operation for process participants and emphasizing their interaction patterns. Furthermore, analyzing agent interaction models can uncover macro-level agent type interaction patterns. These patterns can subsequently inform the design of MAS models and be incorporated into agent-based simulation techniques to more accurately reflect the dynamics of real-world systems.

Tour et al. [40] introduced Agent Miner (AM), the first divide-and-conquer algorithm within the process-aware information systems domain, designed to automatically discover MAS models from event data, following the bottom-up MAS design methodology. AM applies an agent type identification technique to group information on agent instances occurring in event data into agent types. For each identified agent type, an agent model is constructed to represent the behavior of agents that belong to that type. Additionally, an interaction model is derived to capture the communication patterns among agents of various types. Finally, the MAS model is constructed by integrating the discovered agent models with the interaction model. Building on these advancements, agent system mining has found applications in business process simulation. For example, AgentSimulator [19] adopts a resource-first approach, leveraging ABM to uncover agent behaviors and interaction patterns from event data. This information is

used to simulate future process executions, offering insights for process optimization.

Figure 2a and Fig. 2b show the interaction model and the MAS model, respectively, discovered by AM from the BPIC 2014 (incident details) event log. Both models are represented as Petri nets. The interaction model depicts how the three identified agent types coordinate their work within the process. The process begins with an agent of type $A1$ performing initial work. At this point, the process may either terminate or proceed with an agent of type $A2$ or $A3$ performing work, followed by more work by an agent of type $A1$. This behavior can repeat multiple times, as needed, with the process either terminating or continuing with further work by an agent of type $A2$ or $A3$, followed by activities carried out by an agent of type $A1$. The MAS model builds upon the interaction model by refining its transitions to incorporate the corresponding agent models. This refinement operation is formalized by Polyvyanyy et al. [23], refer to Definition 7. It ensures that if the agent models and the interaction model are sound free-choice workflow nets, then the resulting MAS model is a sound workflow net [41].[3] The agent and interaction models constructed by AM are sound free-choice workflow nets. Hence, MAS models constructed by AM are sound. Although the models of individual agents are not explicitly presented, they can be extracted from the MAS model using the interaction model and (the inverse of) the refinement operation. To facilitate this extraction, transitions in the MAS model are labeled with both activity names and the agent types responsible for performing them. As shown in the figure, agents of types $A2$ and $A3$ perform a single activity each time they engage in the process, whereas agents of type $A1$ are responsible for five activities. This labeling approach ensures clear traceability of responsibilities within the MAS model.

In terms of core agent characteristics, the constructed MAS model captures sociability through the representation of interactions between agents in the interaction model. Each agent also captures a degree of autonomy, as its decision-making is guided by the direct transitions defined in its agent type model. The agents can exhibit proactivity by executing the enabled transitions in their agent nets. The agents self-organize via interactions, modeled as token flow between different agent nets, while the emergent behavior can be induced by simulating the MAS model.

The use of Petri nets for constructing MAS models provides several advantages. Petri nets provide a solid mathematical foundation and are well-suited for representing concurrent behaviors of agents engaged in interactions [46]. In addition, Petri nets provide an interpretable visualization of agent behaviors, supporting visual explanations during process simulation. The limitations of Petri net-based MAS models can be addressed in future work by using declarative process mining to discover rules, norms, and goals of agents, enhancing the model's ability to support system-level dynamics.

The MAS model in Fig. 2b achieves entropy-based precision and recall of 0.40 and 0.71, respectively, and consists of 166 nodes and edges.[4] This confirms that AM can

[3] A *workflow net* has a single input place without incoming edges and a single output place without outgoing edges, and each place and transition of the net is on a directed walk from the input place to the output place. A workflow net is *sound* if every its execution can be extended to eventually put a token in the output place, and once a token is deposited in the output place, there are no tokens elsewhere in the net, and every transition can participate in some execution of the net. Note that nets in Figs. 1, 2a and 2b are sound workflow nets.

[4] MAS models explicitly represent information about agents performing activities, enabling the use of this

discover MAS models that are both smaller and more accurate than process models produced by conventional process discovery algorithms. As conventional discovery algorithms do not use information on agents triggering events, transitions in the Petri net in Fig. 1 are labeled solely with activity names. The color coding from Fig. 2 is applied in Fig. 1 to map activities to the corresponding agent types. Notably, activities of agent type $A1$ are dispersed across the process model but are coherent in the MAS model. Hence, the MAS model, in addition to the overall system, supports analysis and improvement of the behaviors of individual process participants. Transitions with the gray background in Fig. 1 refer to activities not present in the MAS model. This omission occurs because AM focuses on capturing the behavioral patterns and interaction logic of agents rather than every activity observed in the event log. Some activities lack the regularity required to form consistent behavioral patterns and are therefore excluded from the MAS model. By filtering out these activities, the MAS model offers a more precise representation of the agents' behaviors and their interactions.

This example highlights the significance of different paradigms in process mining. Viewing event data through an agent-based lens reveals a simple interaction model (Fig. 2a). Refining this model with individual agents' behaviors results in a more accurate and compact representation (Fig. 2b) than the conventional imperative process models (Fig. 1). Although further semantic-preserving simplifications of the nets in Fig. 2 are possible, in this work, we retain them as produced by the original Agent Miner [40] algorithm, leaving such improvements for future work. While the behavior of agent $A1$ appears "tangled" in the MAS model (Fig. 2b), it ultimately encodes simple process constraints. Specifically, when agent $A1$ engages in a case, they begin by executing one of the following activities: "Assignment" (A), "Open" (O), or "Closed" (C). Thereafter, the agent can perform "Assignment", "Closed", "Status Change" (S), and "Caused By CI" (X) any number of times with the following constraint on the order of these activity occurrences. Agent $A1$ cannot perform "Status Change" directly after they perform the "Caused By CI" activity. This flexible behavior is well-suited for representation using a declarative paradigm and can be expressed using the regular expression: $(\mathrm{A}\,|\,\mathrm{O}\,|\,\mathrm{C})(\mathrm{A}\,|\,\mathrm{C}\,|\,\mathrm{X}(?!\mathrm{S})\,|\,\mathrm{S})^*$. Therefore, agent-based models derived from event data not only enhance the conventional approach to process mining but also facilitate the advancement of multi-paradigm process mining.

3 Motivating Example

Different models of agent-based systems can be discovered using different agent types, where each agent type represents a group of agent instances occurring in the event data. In this section, we compare two models of an agent-based system constructed by AM using two distinct sets of agent types. To construct these models, we use a small event log comprising two traces over 96 events extracted from the publicly available BPIC 2015-1 event log. These 96 events refer to eight activities (T_1 to T_8) and six agent

information to compare event logs and MAS models. However, to support the comparison of MAS models with process models discovered from the same event log, here we base the comparison between an event log and a MAS model solely on traces defined as sequences of activities, either recorded in the event log or described as possible executions of the MAS model.

instances (R_1 to R_6) and are listed below. Each event is given as a tuple comprising a case identifier, activity, timestamp, and agent instance.[5]

e_1	$(C_1,T_1,1,R_1)$	e_{17}	$(C_1,T_4,17,R_2)$	e_{33}	$(C_1,T_5,33,R_2)$	e_{49}	$(C_1,T_1,49,R_5)$	e_{65}	$(C_2,T_1,65,R_1)$	e_{81}	$(C_2,T_5,81,R_6)$
e_2	$(C_1,T_1,2,R_1)$	e_{18}	$(C_1,T_4,18,R_2)$	e_{34}	$(C_1,T_5,34,R_2)$	e_{50}	$(C_1,T_1,50,R_5)$	e_{66}	$(C_2,T_1,66,R_1)$	e_{82}	$(C_2,T_5,82,R_6)$
e_3	$(C_1,T_1,3,R_1)$	e_{19}	$(C_1,T_4,19,R_2)$	e_{35}	$(C_1,T_5,35,R_2)$	e_{51}	$(C_1,T_1,51,R_5)$	e_{67}	$(C_2,T_1,67,R_1)$	e_{83}	$(C_2,T_7,83,R_4)$
e_4	$(C_1,T_1,4,R_1)$	e_{20}	$(C_1,T_4,20,R_2)$	e_{36}	$(C_1,T_5,36,R_2)$	e_{52}	$(C_1,T_1,52,R_5)$	e_{68}	$(C_2,T_1,68,R_1)$	e_{84}	$(C_2,T_7,84,R_4)$
e_5	$(C_1,T_2,5,R_1)$	e_{21}	$(C_1,T_4,21,R_2)$	e_{37}	$(C_1,T_5,37,R_2)$	e_{53}	$(C_1,T_1,53,R_5)$	e_{69}	$(C_2,T_1,69,R_6)$	e_{85}	$(C_2,T_7,85,R_4)$
e_6	$(C_1,T_3,6,R_1)$	e_{22}	$(C_1,T_4,22,R_2)$	e_{38}	$(C_1,T_4,38,R_1)$	e_{54}	$(C_1,T_1,54,R_5)$	e_{70}	$(C_2,T_5,70,R_6)$	e_{86}	$(C_2,T_7,86,R_4)$
e_7	$(C_1,T_1,7,R_1)$	e_{23}	$(C_1,T_5,23,R_2)$	e_{39}	$(C_1,T_1,39,R_3)$	e_{55}	$(C_1,T_1,55,R_5)$	e_{71}	$(C_2,T_5,71,R_6)$	e_{87}	$(C_2,T_1,87,R_4)$
e_8	$(C_1,T_1,8,R_1)$	e_{24}	$(C_1,T_5,24,R_2)$	e_{40}	$(C_1,T_1,40,R_3)$	e_{56}	$(C_1,T_7,56,R_4)$	e_{72}	$(C_2,T_5,72,R_6)$	e_{88}	$(C_2,T_7,88,R_4)$
e_9	$(C_1,T_1,9,R_1)$	e_{25}	$(C_1,T_6,25,R_2)$	e_{41}	$(C_1,T_4,41,R_3)$	e_{57}	$(C_1,T_7,57,R_4)$	e_{73}	$(C_2,T_6,73,R_6)$	e_{89}	$(C_2,T_7,89,R_4)$
e_{10}	$(C_1,T_1,10,R_1)$	e_{26}	$(C_1,T_5,26,R_2)$	e_{42}	$(C_1,T_1,42,R_3)$	e_{58}	$(C_1,T_7,58,R_4)$	e_{74}	$(C_2,T_5,74,R_6)$	e_{90}	$(C_2,T_7,90,R_4)$
e_{11}	$(C_1,T_1,11,R_1)$	e_{27}	$(C_1,T_4,27,R_2)$	e_{43}	$(C_1,T_7,43,R_4)$	e_{59}	$(C_1,T_7,59,R_4)$	e_{75}	$(C_2,T_5,75,R_6)$	e_{91}	$(C_2,T_1,91,R_4)$
e_{12}	$(C_1,T_1,12,R_1)$	e_{28}	$(C_1,T_5,28,R_2)$	e_{44}	$(C_1,T_1,44,R_5)$	e_{60}	$(C_2,T_1,60,R_1)$	e_{76}	$(C_2,T_5,76,R_6)$	e_{92}	$(C_2,T_1,92,R_4)$
e_{13}	$(C_1,T_1,13,R_1)$	e_{29}	$(C_1,T_5,29,R_2)$	e_{45}	$(C_1,T_1,45,R_5)$	e_{61}	$(C_2,T_1,61,R_1)$	e_{77}	$(C_2,T_5,77,R_6)$	e_{93}	$(C_2,T_1,93,R_4)$
e_{14}	$(C_1,T_1,14,R_1)$	e_{30}	$(C_1,T_5,30,R_2)$	e_{46}	$(C_1,T_1,46,R_5)$	e_{62}	$(C_2,T_2,62,R_1)$	e_{78}	$(C_2,T_5,78,R_6)$	e_{94}	$(C_2,T_1,94,R_4)$
e_{15}	$(C_1,T_1,15,R_1)$	e_{31}	$(C_1,T_5,31,R_2)$	e_{47}	$(C_1,T_1,47,R_5)$	e_{63}	$(C_2,T_3,63,R_1)$	e_{79}	$(C_2,T_5,79,R_6)$	e_{95}	$(C_2,T_7,95,R_4)$
e_{16}	$(C_1,T_4,16,R_2)$	e_{32}	$(C_1,T_4,32,R_2)$	e_{48}	$(C_1,T_8,48,R_5)$	e_{64}	$(C_2,T_1,64,R_1)$	e_{80}	$(C_2,T_5,80,R_6)$	e_{96}	$(C_2,T_1,96,R_4)$

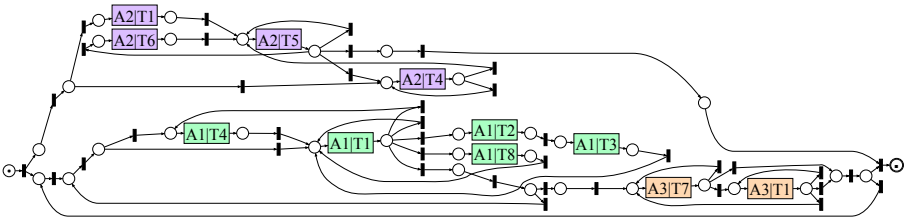
Model I shown in Fig. 3a is an MAS model constructed by AM based on three agent types: $A_1 = \{R_1, R_3, R_5\}$, $A_2 = \{R_2, R_6\}$, and $A_3 = \{R_4\}$. These agent types were identified using the native AM technique detailed in Section 4.1. Model II shown in Fig. 3b is another MAS model discovered by AM using two agent types: $A_4 = \{R_1, R_3, R_4, R_5, R_6\}$ and $A_5 = \{R_2\}$. These two agent types were identified with the help of the BME approach from organizational mining detailed in Section 4.2.4. In both models, again, we label transitions using pairs of agent type and activity name.

Model I and Model II have precision of 0.21 and 0.29, respectively, while both models have a recall of 0.99, where precision and recall are again measured using the entropy-based approach that compares traces defined as sequences of activities. Model I consists of 188 nodes and edges, whereas Model II has 165 nodes and edges. These results highlight the importance of identifying proper agent types, as the choice of agent types can lead to the discovery of a MAS model that not only describes the system more accurately but is also more compact compared to a MAS model derived from a different set of agent types.

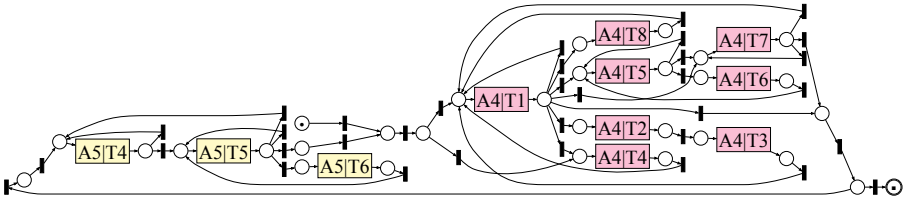
As MAS models integrate multiple agent models into an overall system model, it is interesting to compare them based on the qualities of their agent community structures using modularity. We use Louvain modularity, a widely used method for quantifying the modularity of networks. This measure evaluates the balance between the density of links within communities and the links between communities [3]. Specifically, we measure the modularity of communities formed by activities performed by the same agent types, where links are defined as walks along the edges of the MAS model that do not encounter other activities. Given that directed modularity can offer more accurate insights in scenarios where links are inherently directed [14], as is the case here, we compute both undirected and directed modularity values as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left(W_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j), \tag{1}$$

[5]The motivating example event log can be downloaded from here: https://doi.org/10.26188/24718956.



(a) Model I (precision: 0.21, recall: 0.99, size: 188, undirected modularity: 0.28, directed modularity: 0.37, Gini coefficient: 0.18)



(b) Model II (precision: 0.29, recall: 0.99, size: 165, undirected modularity: 0.26, directed modularity: 0.22, Gini coefficient: 0.23)

Figure 3: Two MAS models constructed from the same input event log using agent types identified by (a) the native Agent Miner approach and (b) the BME approach.

where W_{ij} is the link weight between activity nodes i and j in the MAS model, such that $W_{ij} = 1$ if a link exists and $W_{ij} = 0$ otherwise, k_i is the sum of link weights attached to activity node i, m is the total weight of all links in the MAS model, c_i is the agent type associated with activity node i, and $\delta(c_i, c_j)$ evaluates to one if c_i and c_j refer to the same agent type and it evaluates to zero otherwise. Modularity values range between -0.5 and 1, with larger values reflecting a stronger modular structure and more cohesive groupings into communities. This suggests that agents interact more frequently within their own groups than with agents outside, which is generally desirable for discovering meaningful organizational structures in MASs. The time complexity for computing both undirected and directed modularity is $O(p)$, where p is the number of edges in the MAS model, since each edge needs to be examined once to determine intra- and inter-group connections.

To assess the differences in the distribution of activities among agent types, we use the Gini coefficient [12]:

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n \sum_{i=1}^{n} x_i}, \tag{2}$$

where x_i is the number of activities in the MAS model performed by agent type i, and n is the number of distinct agent types. Gini coefficient values range from zero to one, with lower values indicating a more uniform distribution of activities among agent types, i.e., a more balanced agent workload, which is generally desirable in MASs. In contrast, higher values reflect a greater proportion of activities triggered by a few agent

types, indicating potential workload imbalance. The time complexity for computing the Gini coefficient is $O(q)$, where q is the number of transitions in the MAS model.

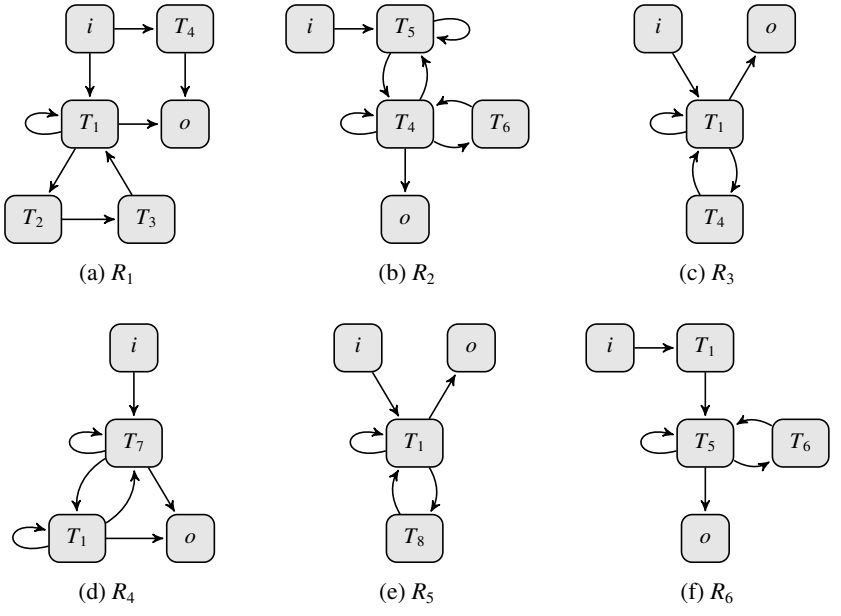
The undirected and directed modularity of Model I are 0.28 and 0.37, respectively, while its Gini coefficient is 0.18. For Model II, the undirected and directed modularity are 0.26 and 0.22, respectively, and the Gini coefficient is 0.23. Model I demonstrates higher modularity than Model II and exhibits a closer-to-uniform distribution of activities across agents. Interestingly, Model II provides a design in which one agent subsumes the skills of another, enabling seamless replacement of the subsumed agent while rendering the subsuming agent a critical, irreplaceable asset. In contrast, Model I, despite some overlap in activities between agent types, promotes a more distinct separation of skills, fostering the specialization of agents in different competencies. This comparison highlights that models optimized for different criteria can offer different insights into system design. It also demonstrates that modularity can provide an additional quality criterion for MAS models, which is a separate optimization target together with precision, recall, and simplicity.

4 Agent Type Identification

This section presents four agent type identification techniques: the technique used in Agent Miner (AM) and three techniques based on organizational mining methods. The Activity Matrix (AMatrix) technique is reused without modifications. The Business Model Enhancement (BME) technique is significantly extended with additional steps to allow agent type identification. Finally, Dominant Cluster Grouping (DCG) is a new technique that only draws broad inspiration from the corresponding organizational mining method. The input to these techniques is an event log, that is, a collection of events, each with four attributes of case identifier, timestamp, activity, and agent, where case identifiers allow grouping the events into traces, each a sequence of activities induced by the total order imposed by the timestamps of the events, and the agent attributes referring to the instances of agents that performed the activities. The output is a set of disjoint agent types, with each type comprising one or more agent instances.

4.1 Agent Miner

The AM algorithm for discovering agent systems from event data incorporates a technique for agent type identification [40]. This technique groups agent instances into agent types by analyzing the similarities between directly-follows graphs (DFGs) that describe their behavior. In a DFG, nodes refer to activities that the agent instance performed in the event log, and arcs encode the directly-follows relation between the activities; that is, the target activity of an arc can directly follow the source activity in some sequence of activities executed by the agent instance, again, as evidenced in the event log. First, for each agent instance, a DFG is constructed. Then, the distance between each pair of DFGs is measured as $1 - (|\mathcal{R}_1 \cap \mathcal{R}_2|)/min(|\mathcal{R}_1|, |\mathcal{R}_2|)$, where $\mathcal{R}_1$ and $\mathcal{R}_2$ are the directly-follows relations of the two DFGs. Finally, if the distance between the DFGs is smaller than or equal to a given threshold t, the instances are grouped in a type.



(a) R_1 (b) R_2 (c) R_3

(d) R_4 (e) R_5 (f) R_6

Figure 4: DFGs describing the behavior of six agent instances from the motivating example event log. T_i are activities and i/o are input/output nodes of the graph.

Consider the motivating example event log from Section 3 and a threshold $t = 0.4$. Figure 4 shows six DFGs, each constructed from the sequences of activities performed by a single agent instance, excluding intermediate activities executed by other agent instances. The DFG of R_1 in Fig. 4a has eight arcs, while the DFG of R_3 in Fig. 4c has five arcs. These DFGs share three arcs: (i, T_1), (T_1, T_1), and (T_1, o), where i and o are the input and output nodes of the DFGs, respectively. Hence, the distance between the DFGs of R_1 and R_3 is $1 - 3/min(5, 8) = 0.4$. Thus, they are grouped in the same agent type. DFGs of R_2 and R_3 do not share arcs. Hence, the distance between them is equal to one, suggesting they should belong to different agent types.

The AM technique identifies three agent types in the example log: $A_1 = \{R_1, R_3, R_5\}$, $A_2 = \{R_2, R_6\}$, and $A_3 = \{R_4\}$. Using these three agent types, the AM algorithm constructs the MAS model in Fig. 3a.

4.2 Techniques Grounded in Organizational Mining

This section reviews existing organizational mining approaches and adapts three suitable approaches for the task of agent type identification.

4.2.1 Suitability of Organizational Mining for Agent Type Identification

A technique for agent type identification should meet three requirements. First, it should be able to group the information about agent instances found in the event log

based on shared characteristics, that is, to perform *clustering* over agent instances.[6] Second, the *input* to such a technique should be an event log, that is, the input event log should contain all the necessary information for the technique to operate effectively. Third, it should ensure that, given an event log, its *output* relates information on a particular agent instance with at most one agent type. The latter is the requirement of AM. Though this requirement can be lifted by defining a dedicated agent type for agent instances belonging to multiple groups, further research is needed to explore why certain organizational mining techniques avoid this simplification and how complex dependencies between agent types impact agent system mining.

We now examine state-of-the-art approaches in organizational mining to evaluate their potential for reuse in agent type identification. Table 1 provides a summary of how these approaches satisfy the three discussed requirements.

Table 1: Organizational mining techniques and agent type identification requirements; "✓"—requirement satisfied, "×"—requirement not satisfied.

Paper references	**Clustering**	**Input**	**Output**
[6, 30–32]	✓	×	×
[1, 7, 8, 28]	×	✓	×
[49]	✓	✓	×
[5, 27, 35]	✓	✓	✓

Schönig et al. [30] use Declarative Process Intermediate Language (DPIL) to discover resource-aware, declarative process models. In an extension of their work, event log pre-processing and model post-processing are applied when discovering resource-aware models [32]. They also combine the DPIL mining framework and organizational rule templates to discover team compositions from logs [31]. In the RALph miner [6], SQL-based technologies and declarative languages are applied to address resource mining and model checking tasks. The techniques presented in these works can be applied to cluster resources according to their roles. However, their inputs contain organizational background knowledge that is often not available in an event log. Although some resource assignment rules that integrate both the control-flow and organizational perspectives can be extracted as outputs, these techniques do not ensure that a single agent instance is assigned to exactly one constructed group.

The technique by Rozinat et al. [28] discovers a colored Petri net that captures a business process from multiple perspectives. Carrera and Jung [7] use Hidden Markov Model (HMM) to construct a probabilistic process model that optimizes resource allocation. They also use HMMs to discover the information diffusion knowledge in processes [8]. The contribution by Alvarez et al. [1] is to apply process discovery over the resource attributes of events to obtain the interactions between resources in a hospital system. These techniques optimize resource allocation, discover multi-perspective processes, and identify information flows but do not group agent instances into types.

[6] In the organizational mining literature, agent instances and types are typically referred to as *resources* and *roles*, respectively. We use *resources* and *roles* when surveying these works but use the agent-oriented terms elsewhere.

While the inputs required for these techniques can be derived from the event log, their outputs do not provide information on the clustering of agent instances.

The OrdinoR technique by Yang et al. [49] supports the discovery of organizational models from event logs. Unlike other techniques that identify information on agent instances or resources based on activity attributes of events, OrdinoR groups agent instances based on values of multiple attributes. Such attributes are obtained from the input event log. However, when considering the output of this technique, a single agent instance may belong to multiple constructed groups.

The works by Burattin et al. [5], Rembert [27], Song and van der Aalst [35] describe clustering techniques. Comprehensive Workflow Mining (CWM) [27] and Activity Matrix (AMatrix) [35] approaches group information on agent instances based on activity and resource attributes of events. The Business Models Enhancement (BME) [5] approach groups activities in a process model under different roles, that is, types of agents that perform them. All the inputs of these approaches can be inferred from the input event log. In addition, the outputs of these approaches satisfy the requirement that the constructed groups are disjoint. Therefore, CWM, AMatrix, and BME methods meet all three requirements and can be reused, with necessary adaptations, in AM for agent type identification.

4.2.2 Dominant Cluster Grouping

In the work on comprehensive workflow mining, Rembert [27] introduces an approach for grouping agent instances based on the common activities they perform. Building on this idea, we propose the DCG technique for agent type identification, described in Algorithm 1. This technique takes as input an event log L over m activities, $\{T_1, \ldots, T_m\}$, and n agent instances, $\{R_1, \ldots, R_n\}$, and a threshold $t \in [0, 1]$. It outputs the set of identified agent types AT, where each type in AT is defined as a set of agent instances.

In the algorithm, $Num(T, L)$ denotes the number of times activity T is executed in event log L, while $Num(R, T, L)$ represents the number of times agent instance R executed activity T in event log L. Let $x = (U, V)$ be a pair where U is a set of agent instances and V is a set of activities. The functions $Rs(x)$ and $Ts(x)$ return the sets of agent instances and activities associated with x, respectively; specifically, $Rs(x) = U$ and $Ts(x) = V$. Given a threshold t, Algorithm 1 in Algorithm 1 construct the sets F and S. Set F contains agent instances that perform a significant share of all executions for at least one activity, while set S consists of pairs relating these agent instances to the activities they frequently execute. For the example event log and a threshold $t = 0.2$, it holds that $F = \{R_1, R_2, R_4, R_5, R_6\}$ and $S = \{(\{R_1\}, \{T_1\}), (\{R_1\}, \{T_2\}), (\{R_1\}, \{T_3\}), (\{R_2\}, \{T_4\}), (\{R_2\}, \{T_5\}), (\{R_2\}, \{T_6\}), (\{R_4\}, \{T_7\}), (\{R_5\}, \{T_1\}), (\{R_5\}, \{T_8\}), (\{R_6\}, \{T_5\}), (\{R_6\}, \{T_6\})\}$.

The repeat loop at Algorithm 1 iteratively merges the pairs in S that share common agent instances or activities. The resulting aggregated pairs represent sets of agent instances associated with the activities they can perform. After the loop completes, the set S for the example log is $\{(\{R_1, R_5\}, \{T_1, T_2, T_3, T_8\}), (\{R_2, R_6\}, \{T_4, T_5, T_6\}), (\{R_4\}, \{T_7\})\}$.

The for loop at Algorithm 1 checks whether the agent instances not included in S, that is, infrequently observed agent instances in L, should be merged with some of the so far constructed pairs in S. For each such infrequent agent instance R_k, see

Algorithm 1: $DCG(L, t)$

Input: Event log L over m activities $\{T_1, \ldots, T_m\}$ and n agent instances $\{R_1, \ldots, R_n\}$, and threshold $t \in [0, 1]$
Output: Set of agent types AT

$AT \leftarrow \emptyset$
$F \leftarrow \{R_i \mid Num(R_i, T_j, L)/Num(T_j, L) \geq t, i \in [1 \,..\, n], j \in [1 \,..\, m]\}$
$S \leftarrow \{(\{R_i\}, \{T_j\}) \mid Num(R_i, T_j, L)/Num(T_j, L) \geq t, i \in [1 \,..\, n], j \in [1 \,..\, m]\}$
repeat until S does not change in the last iteration
  for two distinct pairs x and x' in S **do**
    if $Rs(x) \cap Rs(x') \neq \emptyset$ **or** $Ts(x) \cap Ts(x') \neq \emptyset$ **then**
      $S \leftarrow (S \setminus \{x, x'\}) \cup \{(Rs(x) \cup Rs(x'), Ts(x) \cup Ts(x'))\}$
for $k \in [1 \,..\, n]$ **do**
  if $R_k \notin F$ **then**
    $mf \leftarrow max_{x \in S} \sum_{T \in Ts(x)} Num(R_k, T, L)$
    if $mf > 0$ **then**
      $y \leftarrow argmax_{x \in S} \sum_{T \in Ts(x)} Num(R_k, T, L)$
      $S \leftarrow (S \setminus \{y\}) \cup \{(Rs(y) \cup \{R_k\}, Ts(y))\}$
    else
      $AT \leftarrow AT \cup \{\{R_k\}\}$
for x **in** S **do** $AT \leftarrow AT \cup \{Rs(x)\}$
return AT

Algorithm 1, the algorithm identifies a pair y in S, refer to Algorithm 1, for which R_k, according to L, has performed the largest number of activities from $Ts(y)$. If for this pair y it holds that R_k has performed in L at least one activity in $Ts(y)$ at least once, see Algorithm 1, then R_k is merged with y, see Algorithm 1. Otherwise, R_k defines its own agent type, refer to Algorithm 1. This way, an agent instance that does not perform many activities is grouped with agent instances that perform similar activities. In the example log, the only infrequently occurring agent instance is R_3. Hence, the guard of the if statement at Algorithm 1 evaluates to true only once, when k equals three. Subsequently, R_3 is merged with the pair $(\{R_1, R_5\}, \{T_1, T_2, T_3, T_8\})$ in S, as R_3 has executed activities in $\{T_1, T_2, T_3, T_8\}$ three times in L, which is the largest number of activity executions among all the pairs in S.

Once the for loop at Algorithm 1 completes, the sets of agent instances from all pairs in S are accepted as the resulting agent types, see Algorithm 1. Thus, the DCG algorithm identifies three agent types in the example log: $\{R_1, R_3, R_5\}$, $\{R_2, R_6\}$, and $\{R_4\}$. Notably, for this event log, DCG identifies the same agent types as the AM technique.

4.2.3 Activity Matrix

Song and van der Aalst [35] introduced the AMatrix approach for clustering agent instances based on the similarity of activities they perform. This technique can be directly applied to identify agent types in an event log. First, the approach constructs an activity matrix that summarizes the frequencies of activity executions by each agent instance. Table 2 shows the activity matrix constructed for the motivating example event log. For instance, the matrix shows that agent R_1 has executed activity T_1 twenty times. Next, the Pearson correlation coefficient is computed over the activity frequencies for each pair of agents. The stronger the correlation, the more similar the agents. Then,

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
R_1	20	2	2	1	0	0	0	0
R_2	0	0	0	9	12	1	0	0
R_3	3	0	0	1	0	0	0	0
R_4	6	0	0	0	0	0	13	0
R_5	11	0	0	0	0	0	0	1
R_6	1	0	0	0	11	2	0	0

Table 2: An activity matrix.

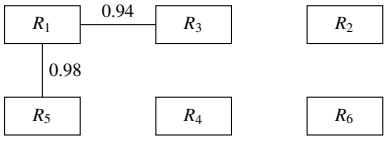


Figure 5: A correlation graph.

a threshold $t \in [0, 1]$, which represents the minimum acceptable correlation strength, is used to filter out pairs of insufficiently similar agent instances. For example, the correlation coefficient between $\vec{R_1} = [20, 2, 2, 1, 0, 0, 0, 0]$ and $\vec{R_3} = [3, 0, 0, 1, 0, 0, 0, 0]$ is 0.94. If $t = 0.9$, R_1 and R_3 belong to the same agent type, as 0.94 exceeds the threshold. Figure 5 illustrates the six agent instances and their strong correlation relationships, where unconnected pairs have correlation coefficients below 0.9. Finally, each connected component in the correlation graph represents a distinct agent type, grouping agents that execute similar activities with similar frequencies.

Applying the presented principles to the example event log, using a threshold $t = 0.9$, the AMatrix technique identifies four agent types: $\{R_1, R_3, R_5\}$, $\{R_2\}$, $\{R_4\}$, and $\{R_6\}$.

4.2.4 Business Models Enhancement

Burattin et al. [5] describe the BME approach for partitioning activities in a process model into agent types responsible for executing these activities. To enable the identification of agent types in an input event log, we introduce two additional steps to complement the original approach. First, we apply Heuristic Miner [47] with a threshold $t_1 \in [0, 1]$ to discover a process model, specifically a DFG, from the input event log.[7] The discovered model serves as input to BME. For instance, the DFG discovered for $t_1 = 0.6$ from the example event log from Section 3 is shown in Fig. 6. Then, we extend BME and post-process the identified agent types to consolidate the results.

The BME approach proceeds in two phases. In the first phase, each edge in the DFG model is assigned a weight representing the frequency of handovers from the agent and activity of the source of the edge to the target agent and activity. For example, consider the edge between T_1 and T_2. To calculate its weight, we collect consecutive event pairs belonging to the same case such that the first event triggered T_1 and the subsequent event triggered T_2. The total number of such event pairs is denoted as n_1. In the example event log, there are two such event pairs: (e_4, e_5) and (e_{61}, e_{62}). Hence, it holds that $n_1 = 2$. Next, the agent pairs triggering these event pairs are recorded, yielding (R_1, R_1) for both pairs. The number of event pairs triggered by the same agent is denoted by n_2. In the running example, it holds that $n_2 = 2$. Next, n_3 represents the number of distinct agents triggering both activities based on the identified event pairs. For the example edge, this yields $n_3 = |\{R_1\}| = 1$. The weight assigned to the edge is then computed as $(n_2+n_3)/(2\times n_1)$. Hence, the weight of the edge between T_1 and T_2 is

[7] We use the publicly available implementation of Heuristic Miner from the `pm4py` package: https://pm4py-source.readthedocs.io/en/stable/_modules/pm4py/algo/discovery/causal/variants/heuristic.html

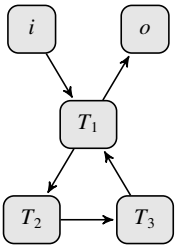


Figure 6: The DFG discovered from the motivating example event log using Heuristic Miner ($t_1 = 0.6$).

equal to $^{(1+2)}/_{(2\times2)} = 0.75$. Edges with a weight below threshold $t_2 \in [0, 1]$ are then removed from the DFG. The activities within each connected component of the model are accepted as performed by the same role. This allows for the assignment of each connected component to a distinct agent type. In the example, setting $t_2 = 0.7$ yields a single connected component comprising the activities $\{T_1, T_2, T_3\}$.

In the second phase, the technique merges the connected components of activities identified in the first phase based on their level of similarity. First, for each connected component, the frequency of agents performing the activities within that component is collected. For example, consider the set of activities $\{T_1, T_2, T_3\}$, which are triggered six times by a single agent, R_1. This information is recorded as an agent-frequency set $S_1 = \{(R_1, 6)\}$ for this connected component. In this example, since only one connected component exists, so the merging step is skipped. Assume there is another connected component with an agent-frequency set $S_2 = \{(R_1, 4)\}$. To calculate the similarity between these two components, two intermediate quantities are computed. First, let n_4 denote the sum of frequencies across both agent sets, which amounts to $n_4 = 6 + 4 = 10$ in this example. Next, n_5 is defined as the sum of the minimum frequencies for agents common to both sets. It follows that $n_5 = \min(6, 4) = 4$ in the example. The similarity between two components is then computed as $^{(2\times n_5)}/_{n_4}$. For the example components, the similarity is $^{(2\times4)}/_{10} = 0.8$. This process is repeated for all pairs of connected components, and components with a similarity score exceeding a given threshold $t_3 \in [0, 1]$ are merged.

The original BME approach produces clusters of activities. In the running example, however, it yields a single activity cluster $\{T_1, T_2, T_3\}$. To address such situations, the last step of our agent type identification technique extends the original BME approach to relate every agent to the activity cluster from which they perform most activities. For each activity cluster, the corresponding set of assigned agents forms an agent type. If an agent does not perform any activity that belongs to any cluster, it is treated as a standalone agent type. Consequently, agents R_1, R_3, R_4, R_5, and R_6 from the example relate to the activity group $\{T_1, T_2, T_3\}$ and hence define one agent type. Agent R_2 defines another agent type because the activities it performs are not in the activity group $\{T_1, T_2, T_3\}$.

A configuration of the BME technique is thus given by a triplet of thresholds (t_1, t_2, t_3). For example, the configuration $(0.6, 0.7, 0.7)$ results in the identification of two agent types: $A_4 = \{R_1, R_3, R_4, R_5, R_6\}$ and $A_5 = \{R_2\}$. Figure 3b presents the MAS model discovered by the AM algorithm using these two agent types.

Table 3: Event logs used in our evaluation.

Event log	Variant freq. filter	#Events	#Cases	#Activities	#Agent inst.
BPIC 2011[9]	10%	3,433	260	168	21
BPIC 2012[10]	80%	225,412	12,214	36	68
BPIC 2013[11]	80%	6,143	1,422	7	593
BPIC 2014 (incident details)[12]	10%	140,101	26,247	33	195
BPIC 2015-1[13]	80%	44,075	1,002	20	21
BPIC 2015-2[14]	80%	36,670	676	24	11
BPIC 2015-3[15]	80%	51,405	1,207	23	10
BPIC 2015-4[16]	80%	39,837	882	26	10
BPIC 2015-5[17]	80%	49,615	958	23	21
BPIC 2020[18]	80%	80,405	6,799	21	8

5 Evaluation

This section evaluates four agent type identification techniques: the native technique of AM [40], DCG [27], AMatrix [35], and BME [5]; see Section 4 for an overview of the techniques. First, we describe the event logs used in the evaluation, the experimental setup, and the implementation of the techniques. Second, we present the results of comparing MAS models and process models discovered using various agent type identification techniques from real-world event logs. Finally, we use synthetic event logs with known ground truth agent types to assess the accuracy of each technique.

5.1 Event logs, Setup, and Implementation

We used ten publicly available real-world Business Process Intelligence Challenge (BPIC) event logs to evaluate and compare the agent type identification techniques discussed in Section 4. We applied *variant frequency filter* to exclude infrequent cases from these event logs, ensuring that the focus is on more representative process behavior. These event logs and variant frequency filters were used in the original evaluation of AM [40]. Tour et al. [40] have identified these event logs as all publicly available event logs shared by the IEEE Task Force on Process Mining in which attributes of events suggest information about agent instances. We *assume* agent-based systems induced these event logs.[8] Specifically, we accept that the resource attributes of events in the selected event logs refer to agents that performed the activities that triggered these events. The characteristics of these ten filtered event logs are summarized in Table 3.

[8] The fact that we do not know if the data originated from agent-based systems makes any positive outcomes extremely valuable, potentially providing evidence for the usefulness of ABM as a paradigm that can be applied for process discovery.

[9] https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

[10] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[11] https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07

[12] https://doi.org/10.4121/uuid:3cfa2260-f5c5-44be-afe1-b70d35288d6d

[13] https://doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17

[14] https://doi.org/10.4121/uuid:63a8435a-077d-4ece-97cd-2c76d394d99c

[15] https://doi.org/10.4121/uuid:ed445cdd-27d5-4d77-a1f7-59fe7360cfbe

[16] https://doi.org/10.4121/uuid:679b11cf-47cd-459e-a6de-9ca614e25985

[17] https://doi.org/10.4121/uuid:b32c6fe5-f212-4286-9774-58dd53511cf8

[18] https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51

As the evaluated technique can be parameterized, refer to Section 4, we chose five different parameters for each technique. When selecting parameters, we aimed to construct MAS models with different numbers of agents for comparison. Two situations should be avoided. Firstly, we aimed to avoid having all the agent-related information found in the event log grouped under the same agent type. Secondly, the situation when every single reference to an agent found in the event log defines its dedicated agent type is undesirable. We fine-tuned the parameters based on the BPIC 2015-1 event log and applied them to all the other evaluated event logs. For the AM approach, we used five thresholds of 0.5, 0.6, 0.8, 0.9, and 1.0. For the DCG technique, we applied the thresholds of 0.5, 0.6, 0.7, 0.8, and 0.9. For the AMatrix method, we chose the thresholds of 0.6, 0.7, 0.8, 0.9, and 0.97. The parameter triplets we used for the BME approach are $(0.4, 0.9, 0.9)$, $(0.6, 0.7, 0.7)$, $(0.7, 0.7, 0.7)$, $(0.8, 0.6, 0.6)$, and $(0.9, 0.5, 0.5)$.

For each identified collection of agent types, we executed AM ten times with ten different parameter configurations (ff_i, th_i), where $ff_i = i \times 0.1$ and $th_i = 1 - i \times 0.1$, $i \in [1\,..\,10]$. The activity frequency filter parameter ff_i and the threshold parameter th_i control the level of detail in the discovered agent models and interaction models, respectively. We used two methods to label transitions in the discovered models: activity-only labeling (AOL) and agent and activity labeling (AAL). For the AOL labeling, transitions in the discovered MAS models are labeled with the names of the activities from the input event log, while for the AAL labeling, a transition label is a pair of an activity from the event log and an identified agent type that performed the activity. Hence, for each event log and agent type identification technique, using AM, we constructed 50 AOL MAS models and 50 AAL MAS models. In addition, for each of the two labeling methods, we used the IM conventional process discovery algorithm to construct 50 (non-agent-based) process models using 50 noise threshold values $th_n = 1 - n \times 0.02$, where $n \in [0\,..\,49]$, hence obtaining 50 AOL process models and 50 AAL process models. We used agent types identified by the BME technique to construct the AAL-labeled IM process models; that is, activity labels of transitions in the models discovered by IM incorporate the information on agent types responsible for the execution of the corresponding activities. Though model activities in conventional process discovery are commonly identified using the activity attributes of events in event logs (AOL labeling), in general, one can use other ways to identify model activities [43]. By identifying model activities as activity-agent pairs (AAL labeling), we aim to incorporate information on agents into process models constructed by IM. Note that an AAL process model cannot be obtained from the AOL process model constructed by IM by using the noise threshold used to discover the AAL model and enhancing the model activities with the information on agent types responsible for performing them. An AAL process model is structurally different from the AOL process model constructed for the same noise threshold, as these two models result from different activities and, thus, different collections of traces.

We used entropy-based precision and recall, and model size to compare the discovered MAS models and process models. Specifically, we evaluated the precision and recall of AOL and AAL MAS and process models concerning event logs where trace activities were identified using corresponding labeling methods. Additionally, undirected and directed modularity, along with the Gini coefficient, were employed to further assess the constructed MAS models. The rationale for using these six quality

Table 4: Best performing techniques based on one-dimensional Welch's *t*-test (AOL).

Event log	Precision	Recall	Size
BPIC 2011	AM	AM,AMatrix,IM	AMatrix,DCG,IM
BPIC 2012	BME	AM,BME,DCG	BME,IM
BPIC 2013	DCG	AM,BME,AMatrix,DCG	IM
BPIC 2014 (inc. details)	AM,BME,AMatrix,DCG	BME,AMatrix	IM
BPIC 2015-1	IM	AM,BME,AMatrix,DCG	IM
BPIC 2015-2	AM,BME,AMatrix	AM,BME,AMatrix,IM	DCG
BPIC 2015-3	IM	AM,BME,AMatrix	IM
BPIC 2015-4	BME,AMatrix	AM,BME,AMatrix	IM
BPIC 2015-5	AM,BME,AMatrix,DCG,IM	BME,AMatrix	IM
BPIC 2020	AM,BME,AMatrix,DCG,IM	AM,BME,AMatrix,DCG,IM	AMatrix

Table 5: Best performing techniques based on one-dimensional Welch's *t*-test (AAL).

Event log	Precision	Recall	Size
BPIC 2011	AM	AM,BME,AMatrix	AM,BME,AMatrix,DCG
BPIC 2012	BME	BME,DCG	IM
BPIC 2013	DCG	AM,BME,AMatrix,DCG	IM
BPIC 2014 (inc. details)	BME,IM	AM,BME,AMatrix	IM
BPIC 2015-1	AMatrix,IM	BME,AMatrix,DCG	IM
BPIC 2015-2	BME,IM	AM,BME,AMatrix,DCG	IM
BPIC 2015-3	IM	BME,AMatrix	IM
BPIC 2015-4	BME,AMatrix	AM,BME,AMatrix,DCG	IM
BPIC 2015-5	BME,AMatrix,IM	BME,AMatrix	IM
BPIC 2020	AM,BME,AMatrix,DCG,IM	AM,BME,AMatrix,DCG,IM	BME,AMatrix,DCG,IM

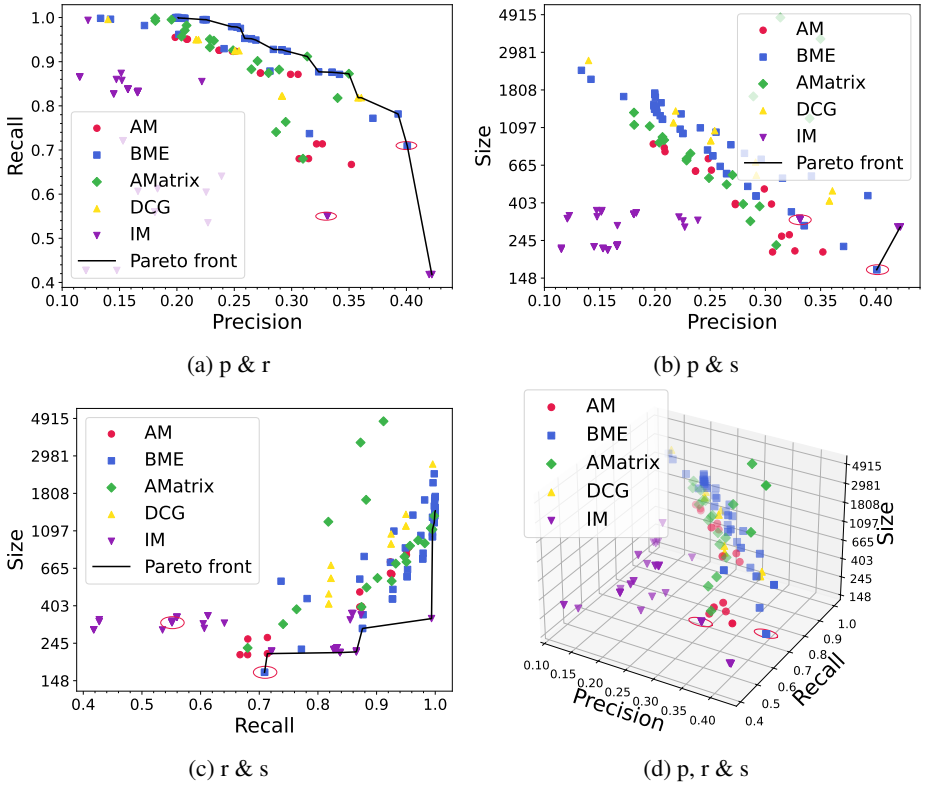
measures is discussed in Section 3. All the evaluated agent type identification techniques have been implemented and made publicly available, along with the code used for the experiments.[19]

5.2 Models Discovered From Real-World Event Logs

The agent types identified based on the BME technique consistently lead to the discovery of better models over most quality measures and event logs, while the AM agent type identification technique supports the discovery of good-quality modular systems. In what follows, we support this claim with the experimental evidence.

We grouped MAS models discovered using AM into four groups based on the agent type identification technique used to construct them, while process models discovered by IM formed the fifth group. In every pair of groups, the models are unpaired, exhibiting differing variances. Therefore, to identify superior groups of models, we performed Welch's *t*-test using the null hypothesis that the performance of models in different groups is the same, for a 95% confidence level, separately for precision, recall, and size. Because each quality criterion was addressed separately, we refer to this comparison as a one-dimensional (1D) comparison. Tables 4 and 5 report the results of the tests, that is, techniques that discover the most accurate, in terms of precision and recall, and simple models, either MAS models or process models, for the AOL and AAL labeling methods, respectively. Some cells in the table list more than one technique,

[19] https://github.com/QINGTANS/agent-type-identification



(a) p & r

(b) p & s

(c) r & s

(d) p, r & s

Figure 7: Pareto fronts for models (AOL) discovered from the BPIC 2014 (incident details) event log, where p, r, and s refer to precision, recall, and size, respectively.

signifying that these techniques are significantly better than other approaches in the corresponding quality criterion, while the models stemming from the listed techniques have no obvious differences. For instance, the size quality dimension for the BPIC 2012 event log shows BME and IM, stating that these techniques lead to the discovery of smaller models. Note that some constructed agent-based models are large due to specific configurations based on the evaluated parameters that aim to cover the parameter space uniformly. However, as it will be clear based on multi-objective, that is, two- (2D) and three-dimensional (3D) analyses, many MAS models are small and accurate. For the AOL labeling, BME and AMatrix techniques lead to the discovery of superior MAS models more often, 16 times for each technique, while conventional IM process models have better quality characteristics 15 times. BME and IM models also have superior qualities in the case of AAL labeling, 18 and 16 times, respectively.

Figures 7a to 7c show three types of 2D Pareto fronts for the MAS models with AOL labeling constructed from the BPIC 2014 (incident details) event log. In Fig. 7a, the Pareto front gravitates toward the top-right corner; see the line segments that connect the data points on the Pareto front in the figure. Models on this Pareto front

Table 6: Techniques with the majority of discovered models included in the 2D Pareto fronts; p, r, and s refer to precision, recall, and size, respectively.

Event log	AOL			AAL		
	p & r	p & s	r & s	p & r	p & s	r & s
BPIC 2011	AM	AM	AM	AM	AM	AM
BPIC 2012	BME	IM	BME	BME	BME, IM	BME
BPIC 2013	DCG	AMatrix, DCG	DCG	DCG	AMatrix, DCG	DCG
BPIC 2014 (inc. d.)	BME	BME, IM	BME	BME	IM	BME
BPIC 2015-1	BME	IM	AMatrix	AMatrix	AMatrix	BME, AMatrix
BPIC 2015-2	AMatrix, IM	IM	IM	BME	IM	BME
BPIC 2015-3	AMatrix, IM	IM	BME	AMatrix	IM	AMatrix
BPIC 2015-4	BME	AMatrix	IM	AMatrix	AMatrix	IM
BPIC 2015-5	BME	BME	BME	BME	BME, AMatrix	BME, IM
BPIC 2020	IM	IM	AM, IM	AM	IM	AM

dominate, that is, perform better than, other models with respect to precision and recall. There are thirty-two MAS models on the precision-recall Pareto front, of which twenty-six were constructed using the agent types identified using the BME technique. The other six models are MAS models constructed using AMatrix and DCG agent types and IM process models, while there are no MAS models based on AM agent types on this Pareto front. In Figs. 7b and 7c, the Pareto fronts gravitate toward the bottom-right corners of the plots, as smaller model sizes are preferable. The precision-size front has two BME and two IM models. The recall-size front contains nineteen models. Thirteen of these models are constructed using the BME agent types. Thus, for the BPIC 2014 (incident details) event log, BME has the most models on all 2D Pareto fronts; that is, it leads to the construction of more high-performing models than other techniques. Table 6 reports the techniques that generated the most models on the 2D Pareto fronts for all the evaluated event logs. Again, BME-induced MAS models performed best among the agent-based techniques. Specifically, they resulted in superior models more often on 2D Pareto fronts 11 times, both for AOL and AAL labeled models, performing particularly well for the BPIC 2012, 2014, and 2015-5 event logs. The IM models were identified as superior 12 and 7 times for the AOL and AAL labeling methods. We conclude that BME performs best on 2D Pareto fronts.

We compare the areas bounded by 2D Pareto fronts of the evaluated techniques. The technique with the largest area generates the most diverse models, thus supporting human experts in choosing the most suitable MAS models according to their needs. We compute the areas below the fronts for the precision-recall plots, while for the other two types of plots, we compute the areas above the fronts. We restrict the ranges of quality measures considered to ignore the impact of low-quality models when calculating the areas. Specifically, we ignore models with precision and recall below 0.2 and 0.7, respectively, and models of size larger than 500.

Tables 7a to 7c report the areas for the AOL labeled models. Process models discovered by IM have the largest areas for most event logs for the precision-recall and precision-size fronts. In contrast, MAS models induced by BME agent types result in the largest areas for most event logs for the recall-size fronts and have better performance for the other two types of 2D fronts compared to the other agent-based techniques. This observation implies that models on the BME Pareto fronts are not pressed

Table 7: Areas and volumes bounded by 2D and 3D Pareto fronts (AOL); p, r, and s refer to precision, recall, and size, respectively.

Log	AM	BME	AMatrix	DCG	IM
2011	**0.071**	0	0	0	0
2012	0	**0.058**	0	0.013	0.056
2013	0.079	0.104	0.102	0.102	**0.110**
2014	0.020	**0.037**	0.016	0.019	0.003
2015-1	0.037	**0.069**	0.065	0.029	0.067
2015-2	0	0	0	0	**0.009**
2015-3	0.026	0.024	0.025	0	**0.034**
2015-4	0.003	0.006	0.006	0	**0.009**
2015-5	0.003	**0.017**	0.006	0	0.007
2020	0	0	0	0	**0.018**

(a) p & r

Log	AM	BME	AMatrix	DCG	IM
2011	**0.115**	0	0	0	0
2012	0	**0.139**	0	0.044	0.076
2013	0.230	0.331	0.310	0.310	**0.342**
2014	0.073	**0.134**	0.034	0.027	0.007
2015-1	0.093	0.196	0.206	0.078	**0.214**
2015-2	0	0	0	0	**0.034**
2015-3	0.067	0.075	0.076	0	**0.170**
2015-4	0.014	0.029	0.029	0	**0.113**
2015-5	0.009	**0.069**	0.047	0	0.034
2020	0	0	0	0	**0.080**

(b) p & s

Log	AM	BME	AMatrix	DCG	IM
2011	**0.069**	0	0	0	0
2012	0	**0.149**	0	0.083	0.100
2013	0.206	**0.236**	**0.236**	**0.236**	0.235
2014	0.063	**0.104**	0.046	0.020	0.051
2015-1	0.194	0.234	**0.251**	0.170	0.203
2015-2	0	0	0	0	**0.081**
2015-3	0.162	0.207	**0.211**	0	0.131
2015-4	0.067	**0.084**	0.082	0	0.047
2015-5	0.117	**0.124**	0.088	0	0.066
2020	0	0	0	0	**0.084**

(c) r & s

Log	AM	BME	AMatrix	DCG	IM
2011	**0.024**	0	0	0	0
2012	0	**0.037**	0	0.007	0.025
2013	0.059	0.086	0.085	0.085	**0.091**
2014	0.011	**0.025**	0.006	0.003	0.001
2015-1	0.025	**0.057**	**0.057**	0.020	0.054
2015-2	0	0	0	0	**0.005**
2015-3	0.017	0.020	0.020	0	**0.027**
2015-4	0.002	0.004	0.004	0	**0.007**
2015-5	0.002	**0.013**	0.004	0	0.004
2020	0	0	0	0	**0.010**

(d) p, r & s

together, confirming that BME agent types can lead to good models of diverse quality characteristics. Similar conclusions can be drawn based on the areas induced by the 2D Pareto fronts of AAL labeled models, refer to Tables 8a to 8c.

A model with two good characteristics can fall short on the third one. To achieve solid conclusions, we study 3D Pareto surfaces comprising the best-performing models that dominate over all three quality dimensions. In Fig. 7d, the models with better precision, better recall, and smaller size are located on the bottom right part of the plot. The 3D Pareto surface in Fig. 7d is defined by 47 models, with most models discovered by AM using BME agent types.

For different event logs, Tables 9 and 10 report the numbers of models on the 3D Pareto surfaces discovered by each evaluated technique; the number of models for the winning technique is highlighted in boldface. We analyze 3D Pareto surfaces that, in combination with precision and recall, use model size (s), directed (d) and undirected (u) modularity, and Gini coefficients (g) as measures of model simplicity.

The BME-induced MAS models are the majority of models on the model size surfaces for AOL and AAL models across the event logs. Moreover, only BME models are present on all the model-size surfaces. Tables 7d and 8d also confirm that BME Pareto surfaces are often more voluminous than surfaces induced by other agent identification techniques and are comparable to the volumes induced by the IM surfaces. Finally, with respect to modularity, the original agent type identification technique of AM demonstrates superior performance across most datasets, while BME provides a more uniform distribution of activities across the agent models.

Table 8: Areas and volumes bounded by 2D and 3D Pareto fronts (AAL); p, r, and s refer to precision, recall, and size, respectively.

Log	AM	BME	AMatrix	DCG	IM
2011	**0.071**	0	0	0	0
2012	0	**0.054**	0	0.010	**0.054**
2013	0.042	0.102	0.102	0.102	**0.110**
2014	0.001	**0.025**	0.011	0	0.006
2015-1	0.010	0.055	**0.063**	0.007	0.045
2015-2	0	0	0	0	**0.009**
2015-3	0	0.024	0.025	0	**0.035**
2015-4	0.003	0.006	0.006	0	**0.009**
2015-5	0	**0.006**	0.005	0	0.005
2020	0	0	0	0	**0.011**

(a) p & r

Log	AM	BME	AMatrix	DCG	IM
2011	**0.115**	0	0	0	0
2012	0	**0.139**	0	0.044	0.067
2013	0.144	0.310	0.310	0.310	**0.342**
2014	0.001	**0.048**	0.013	0	0.014
2015-1	0.028	0.170	**0.206**	0.017	0.116
2015-2	0	0	0	0	**0.033**
2015-3	0	0.075	0.076	0	**0.170**
2015-4	0.014	0.029	0.029	0	**0.113**
2015-5	0	**0.043**	**0.043**	0	0.025
2020	0	0	0	0	**0.067**

(b) p & s

Log	AM	BME	AMatrix	DCG	IM
2011	**0.069**	0	0	0	0
2012	0	**0.140**	0	0.062	0.089
2013	0.194	**0.236**	**0.236**	**0.236**	0.235
2014	0.033	**0.080**	0.036	0	0.050
2015-1	0.170	0.229	**0.250**	0.177	0.172
2015-2	0	0	0	0	**0.073**
2015-3	0	0.206	**0.211**	0	0.132
2015-4	0.066	0.082	**0.083**	0	0.048
2015-5	0	**0.096**	0.084	0	0.058
2020	0	0	0	0	**0.045**

(c) r & s

Log	AM	BME	AMatrix	DCG	IM
2011	**0.024**	0	0	0	0
2012	0	**0.035**	0	0.006	0.022
2013	0.032	0.084	0.084	0.084	**0.091**
2014	0	**0.010**	0.002	0	0.002
2015-1	0.007	0.046	**0.055**	0.004	0.030
2015-2	0	0	0	0	**0.005**
2015-3	0	0.020	0.020	0	**0.028**
2015-4	0.002	0.004	0.004	0	**0.007**
2015-5	0	**0.005**	0.004	0	0.003
2020	0	0	0	0	**0.005**

(d) p, r & s

To emphasize the added value of discovering agent-based models from event logs, we look again at the two models discovered from the BPIC 2014 (incident details) event log discussed in Section 2. The first model is the process model constructed by IM shown in Fig. 1. The second model is the MAS model constructed by AM using agent types identified using the BME technique shown in Fig. 2b. These models reside on the 3D Pareto surfaces composed from models constructed using the respective discovery techniques. That is, they are superior models according to these discovery techniques. In Fig. 7, we emphasize the quality characteristics of these two models with ellipses with red borderlines. The MAS model resides on the global 3D Pareto surface constructed from models discovered by all the evaluated techniques. Recall that this MAS model outperforms the process model in all three quality criteria of precision, recall, and simplicity; refer to Section 2 for details. Notably, the process model in Fig. 1 is the "closest," in terms of the three quality characteristics, to this MAS model among all the 50 discovered process models using IM stemming from the systematic exploration of its parameter space. This observation provides strong evidence that the agent-based modeling paradigm is an interesting paradigm to be used for automatically constructing models from event data of business processes, as MAS models discovered by AM complement process models constructed by a state-of-the-art conventional process discovery algorithm. Interestingly, the smallest of the 50 models constructed by IM from this event log contains 216 nodes and edges, confirming that a discovered MAS model can be smaller than discovered process models while also providing additional information about the interaction structure of the individual agents participating

Table 9: Numbers of models on 3D Pareto surfaces (AOL); p, r, s, d, u, and g refer to precision, recall, size, directed modularity, undirected modularity, and Gini coefficient, respectively.

Log	Number of models (AOL)																
	p-r-s					p-r-d				p-r-u				p-r-g			
	AM	BME	AMatrix	DCG	IM	AM	BME	AMatrix	DCG	AM	BME	AMatrix	DCG	AM	BME	AMatrix	DCG
2011	**9**	1	0	0	2	**10**	8	6	0	**10**	8	6	0	**9**	7	3	0
2012	1	**7**	0	0	4	0	6	6	**13**	1	5	6	**10**	0	**5**	0	0
2013	0	3	6	**10**	6	**11**	9	7	2	**11**	2	8	4	11	**12**	6	2
2014	1	**29**	4	3	10	8	**24**	**16**	6	7	**25**	**16**	6	**9**	**25**	7	6
2015-1	1	**7**	**7**	0	5	**10**	**7**	4	5	**12**	**7**	4	5	3	6	**8**	0
2015-2	1	2	3	0	**7**	**10**	5	3	0	**10**	**11**	4	0	0	**5**	3	0
2015-3	2	3	5	0	**6**	**6**	3	4	0	**8**	3	5	0	2	1	**5**	0
2015-4	2	**7**	4	0	5	7	**9**	4	1	7	8	**9**	1	2	1	**5**	0
2015-5	3	**13**	7	0	**14**	5	**13**	8	0	5	**17**	6	0	5	**17**	7	0
2020	6	4	0	3	**8**	**14**	4	5	7	**14**	5	8	7	**9**	5	0	3

Table 10: Numbers of models on 3D Pareto surfaces (AAL); p, r, s, d, u, and g refer to precision, recall, size, directed modularity, undirected modularity, and Gini coefficient, respectively.

Log	Number of models (AAL)																
	p-r-s					p-r-d				p-r-u				p-r-g			
	AM	BME	AMatrix	DCG	IM	AM	BME	AMatrix	DCG	AM	BME	AMatrix	DCG	AM	BME	AMatrix	DCG
2011	**9**	1	1	0	0	**10**	7	7	1	**10**	7	7	1	**10**	6	1	1
2012	0	**9**	0	0	6	0	12	4	**18**	0	8	6	**16**	0	**8**	0	0
2013	0	1	6	**11**	5	**15**	11	5	4	**15**	1	8	5	**15**	16	3	4
2014	1	**26**	6	0	13	2	15	**19**	0	0	15	**19**	0	**17**	**27**	9	0
2015-1	0	4	**6**	2	0	6	5	**7**	**7**	**11**	5	7	7	0	4	**10**	0
2015-2	0	**8**	1	0	6	**10**	9	1	2	**10**	**10**	1	5	0	**8**	0	0
2015-3	0	1	**8**	0	4	6	7	**8**	0	**10**	7	9	0	0	0	**8**	0
2015-4	0	3	6	0	**7**	8	8	**9**	3	**11**	4	10	3	0	3	**5**	0
2015-5	0	10	4	0	**12**	9	**16**	7	4	9	**23**	6	4	8	**21**	6	0
2020	**8**	4	0	2	4	**9**	5	7	**9**	**9**	5	**15**	7	**6**	3	0	1

in the process and their detailed behaviors.

5.3 Agent Types Identified Based on Synthetic Event Logs

AMatrix consistently identifies agent types closest to the ground truth types, as confirmed for multiple event logs and similarity measures, while the BME and DCG techniques are best in rediscovering agent types in specific contexts. Next, we justify this claim with empirical evidence.

In addition to evaluating the quality of MAS models generated using different agent type identification techniques, we assessed the accuracy of the identified agent types. Specifically, we compared the agent types identified by the four evaluated techniques against ground truth agent types. To facilitate this analysis, we generated synthetic event logs using MAS models constructed by AM. These constructed MAS models provide ground truth agent types. We then checked how well the evaluated agent type identification techniques can infer those ground truth types.

We used the ten real-world event logs listed in Table 3 to construct MAS mod-

els. For each event log, four agent type identification techniques (AM, BME, AMatrix, and DCG) were applied. Each technique was evaluated with five different threshold configurations, as described in Section 5.2, resulting in 20 collections of agent types. For each collection, AM was executed three times with parameter configurations of $(0.1, 0.9)$, $(0.5, 0.5)$, and $(0.9, 0.1)$, generating 60 MAS models per event log. Subsequently, three event logs were generated using each MAS model, each adhering to one of three policies for assigning agent instances to events, as detailed below; note that the assignment of agent instances to agent attributes of events was based on the agent types responsible for performing activities, as identified in the discovered MAS model.

Policy 1: The agent attribute of each event triggered by an activity performed by an agent of some type is assigned a random agent instance of that type.

Policy 2: If multiple consecutive events of a case are triggered by activities performed by agents of the same type, the agent attributes of all these events are assigned a random agent instance of that type.

Policy 3: If multiple events of a case are triggered by activities performed by agents of the same type, the agent attributes of all these events are assigned a random agent instance of that type.

Each generated event log comprises 1,000 cases, with a maximum number of 200 events per case. If a MAS model generated more than 200 events for some case, that case was discarded, and another case was generated. Based on the number of cases and events specified, we ensure that the generated event logs encompass important knowledge about the system while minimizing the generation of excessive repetitive activity patterns. Finally, we used five agent instances of each type to assign agent attributes of events in the generated event logs. The code we used to construct the MAS models and generate event logs is publicly available.[20]

For each generated event log, we applied all four evaluated agent type identification techniques, each using ten parameter configurations. Here, we explore a broader range of thresholds to achieve a better coverage of the search space. For the AM, DCG, and AMatrix techniques, we used thresholds $th_j = 0.5 + j \times 0.05$, $j \in [1 \,..\, 10]$. Finally, for the BME technique, we used these parameter triplets: $(0.5, 0.9, 0.9)$, $(0.5, 0.5, 0.5)$, $(0.6, 0.8, 0.8)$, $(0.6, 0.6, 0.6)$, $(0.7, 0.7, 0.7)$, $(0.8, 0.6, 0.6)$, $(0.8, 0.7, 0.7)$, $(0.8, 0.8, 0.8)$, $(0.9, 0.5, 0.5)$, and $(0.9, 0.9, 0.9)$.

As there are multiple measures for comparing partitions of a set, and no measure is universally preferred or is more suitable for our setting, we employed four commonly used measures to evaluate the agreement between the ground truth and the identified agent types. The Rand Index (RI) quantifies the similarity between two partitions of a set [26]. The Normalized Mutual Information (NMI) quantifies the mutual dependence or shared information between partitions [10], and is widely applied in community detection within social network analysis [21]. Similarly, the Jaccard Index (JI) measures the similarity between sets by treating each pattern as an element of a set [15]. Finally, the Fowlkes-Mallows Index (FMI) provides a numerical assessment of the similarity between two hierarchical clusterings [16]. The similarity scores for all these measures range from zero to one. Higher similarity scores consistently indicate a closer match

[20] https://doi.org/10.26188/28478069

between the experimental results and the ground truth.

Let T and I be ground truth agent types and identified agent types, respectively, each given as a partition of a set of agent instances, where a group of agent instances in a partition defines an agent type. When grouping agent instances into types, an agent identification technique makes decisions on which pairs of agent instances to group in the same agent type and which to attribute to different agent types. Consequently, one can establish a quality of agent type identification in terms of correct and wrong such decisions. Let *TP* (true positives) be the number of pairs of agent instances that are in the same agent type in T and in the same agent type in I. Let *TN* (true negatives) be the number of pairs of agent instances that are in different agent types in T and in different agent types in I. Let *FP* (false positives) be the number of pairs of agent instances that are in different agent types in T and in the same agent type in I. Finally, let *FN* (false negatives) be the number of pairs of agent instances that are in the same agent type in T and in different agent types in I.

Table 11 shows the definitions of the discussed partition similarity measures. Note that RI, JI, and FMI are captured in terms of correct and wrong decisions made by the agent type identification technique. For instance, let $T = \{\{R_1, R_2\}, \{R_3, R_4\}\}$ and $I = \{\{R_1, R_2\}, \{R_3\}, \{R_4\}\}$. In this example, it holds that $TP = |\{\{R_1, R_2\}\}| = 1$, $TN = |\{\{R_1, R_3\}, \{R_1, R_4\}, \{R_2, R_3\}, \{R_2, R_4\}\}| = 4$, $FP = 0$, and $FN = |\{\{R_3, R_4\}\}| = 1$. Based on these terms, Table 11 provides similarity scores between example partitions T and I for the RI, JI, and FMI measures.

The NMI measure uses the mutual information (MI) between T and I that measures how much knowing the agent type assignment of one partition reduces the uncertainty of the other, and normalizes it to ensure the score lies between zero and one. The definition of NMI is given in Table 11. Let $T = \{T_1, T_2, \ldots, T_k\}$ and $I = \{I_1, I_2, \ldots, I_l\}$ be two partitions. Then, the mutual information between T and I is defined as:

$$MI(T, I) = \sum_{i=1}^{k} \sum_{j=1}^{l} P(T_i \cap I_j) \log \left(\frac{P(T_i \cap I_j)}{P(T_i)P(I_j)} \right),$$

where $P(T_i)$ is the probability of an element belonging to group T_i, $P(I_j)$ is the probability of an element belonging to group I_j, and $P(T_i \cap I_j)$ is the probability of an element belonging to both groups T_i and I_j. To normalize the mutual information, entropies $H(T)$ and $H(I)$ of partitions T and I are used, where $H(T) = -\sum_{i=1}^{k} P(T_i) \log P(T_i)$ and $H(I) = -\sum_{j=1}^{l} P(I_j) \log P(I_j)$. Consequently, the NMI score for the example partitions T and I is 0.800, also reported in Table 11.

Based on the measures discussed, Table 12 presents the best-performing agent type identification techniques—those that identify agent types most similar to the ground truth. For each event log, we computed similarity scores between the identified and ground truth agent types. The technique with the highest average similarity score is reported as the best performer in the table.

Overall, AMatrix demonstrates superior performance across nearly all event logs and measures, accurately grouping agent instances into their respective types. BME and DCG achieve the highest accuracy for certain event logs when evaluated using the NMI and JI measures, highlighting their strength in specific contexts.

Table 11: Definitions and examples of measures for comparing collections of agent types given as partitions of a set of agent instances.

Measure	Definition	Example similarity
Rand Index	$\frac{TP+TN}{TP+FP+FN+TN}$	0.833
Jaccard Index	$\frac{TP}{TP+FP+FN}$	0.500
Fowlkes-Mallows Index	$\frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$	0.707
Normalized Mutual Information	$\frac{2MI(T,I)}{H(T)+H(I)}$	0.800

Table 12: Techniques that identify agent types closest to the ground truth based on different evaluation measures.

Event log	RI	NMI	JI	FMI
BPIC 2011	AMatrix	AMatrix	DCG	AMatrix
BPIC 2012	AMatrix	AMatrix	BME	AMatrix
BPIC 2013	AMatrix	BME	AMatrix	AMatrix
BPIC 2014 (inc. details)	AMatrix	AMatrix	AMatrix	AMatrix
BPIC 2015-1	AMatrix	AMatrix	AMatrix	AMatrix
BPIC 2015-2	AMatrix	AMatrix	AMatrix	AMatrix
BPIC 2015-3	AMatrix	BME	AMatrix	AMatrix
BPIC 2015-4	AMatrix	BME	DCG	AMatrix
BPIC 2015-5	AMatrix	AMatrix	AMatrix	AMatrix
BPIC 2020	AMatrix	AMatrix	AMatrix	AMatrix

5.4 Limitations and Threats to Validity

It is important to discuss the limitations of this work and the potential threats to the validity of the results, which we do next.

Evaluating model quality in process mining is known to be computationally expensive. Due to this challenge and the broad scope of our analysis, we excluded models for which computing precision and recall exceeded four hours. As a result, 23.2% of the constructed models were excluded from evaluation. Though these models are typically large and, hence, are rarely used in practice by process analysts, their exclusion may affect the validity of our conclusions. However, as the entropy-based precision and recall measurements we used satisfy all known properties for this class of model quality measures [37], they ensure an unbiased basis for our conclusions.

To compare the performance of various algorithms on a given event log, we discovered 50 models using each algorithm. We then compared algorithm performance based on the models of superior characteristics they produced. This indirect comparison poses a threat to validity, as different parameter settings could yield different models and, consequently, different conclusions. To mitigate this, we selected configuration parameters designed to cover the relevant parameter space uniformly. While this approach does not guarantee the best-performing models for each technique, it aims to reveal general trends in model quality. The decision to discover 50 models was guided by the need to balance experiment depth with runtime feasibility. This setup allowed us to complete all experiments within six weeks on a Linux server with a 64-bit processor

architecture, 20 cores, and 80GB RAM.

Again, to keep the runtimes of the experiments within a reasonable timeframe, we constructed AAL-labeled IM process models using only agent types identified by the BME technique. The use of alternative agent type identification techniques could lead to different conclusions. We selected the BME technique because it consistently led to the construction of superior MAS models.

The experiments were conducted on ten industrial event logs. Though they stem from a range of domains, to improve the generalizability of the results, it is desirable to extend the analysis to more datasets. One can investigate the relationship between the accuracy of the identified agent types and the quality of the resulting MAS models constructed from them. Such studies are postponed for future work.

The conclusion that AMatrix identifies ground truth agent types is based on synthetic data. Specifically, the event logs used for agent type identification were generated by MAS models that were themselves derived from real-world business process event logs. However, since these MAS models are approximations rather than true representations of agent-based systems within organizations, the generated event logs do not reflect actual historical records of agent activities at organizations. It is unclear if the same conclusion will hold to agent type identification over industrial logs.

Another limitation of the current agent-type identification techniques is their assumption that each agent can be assigned to only one agent type. However, in real-world scenarios, agents may play multiple roles and perform different types of activities depending on the roles they play. Therefore, this constraint reduces the flexibility and generalizability of the proposed techniques. Future work may focus on developing agent type identification techniques that support multi-role behaviors for individual agents. This includes modeling the conditions under which an agent plays a specific role, as well as capturing dynamic role adaption based on the agent's current state.

A potentially limiting design choice of our approach is the use of Petri nets to represent MASs. While Petri nets are widely used in process mining as well-understood and expressive abstractions for modeling and analyzing workflows, they are fundamentally discrete and purely reactive. As a result, they are limited in their ability to capture specific characteristics of agents in MASs, such as autonomy and proactivity. Since agent system mining is in its early stages, it relies on many modeling and evaluation methods from process mining. To ensure consistency and allow for comparative assessments using established quality measures, such as precision, recall, and simplicity, we rely on Petri nets as the underlying formalism in this study. As part of future work, one can investigate the application of declarative process mining techniques to discover MAS models, which can generate more flexible specifications that better reflect the rules, norms, and goals of agents and MASs.

Finally, the recent introduction of Agent System Event Data [34], which describes business processes as interactions of agents, shows the potential to collect additional attributes such as organization and role information from the agent system. These additional attributes facilitate the development of top-down agent system mining techniques, where the mining process begins by identifying organizational behaviors and goals, followed by the mining of role-level and individual agent behaviors. This approach supports the analysis of macro-level emergent behaviors from organization-level interactions, and it can provide insights for designing self-organizing systems by

uncovering governing rules at both the organizational and role levels.

6 Conclusions

This article demonstrates that agent-based modeling is an effective approach for automatically discovering simple and accurate models from event data generated during business process execution. These models enhance data-driven business process analysis in two ways. First, they often achieve superior performance according to standard process mining criteria, such as precision, recall, and model size, ensuring they succinctly and accurately represent the underlying processes. Second, agent-based models provide a fresh perspective by focusing on the agents performing process activities and their interactions, enabling detailed analysis of individual participant behaviors and their interaction patterns. These claims are validated through experiments using multiple event logs from real-world business processes.

Identifying agent types, which are groups of agents that perform similar tasks and can be represented by a single agent model, is a crucial step in discovering agent-based models from event data. The challenge of uncovering relationships between process participants from event data is a key focus of organizational mining, a subfield of process mining. Therefore, we reviewed existing organizational mining techniques and adapted three suitable ones, namely DCG [27], AMatrix [35], and BME [5], to solve the agent type identification problem. We then evaluated the Agent Miner algorithm [40] for discovering agent-based models using the original and three newly adapted agent type identification techniques to determine which method produces better models and accurately identifies ground truth agent types.

The BME technique consistently identifies agent types that lead to the discovery of simple, accurate, and versatile agent-based models. In contrast, AMatrix often reliably identifies agent types that closely align with the ground truth, while BME and DCG perform best at rediscovering ground truth agent types for specific datasets and ways of comparing learned agent types with the ground truth. Since agent-based models can be evaluated based on their modularity, defined by the composition and structure of their agent models, we assessed this property across the techniques. Our analysis revealed that the original agent type identification technique of Agent Miner most frequently produces good-quality modular models.

Interestingly, in our experiments, identifying agent types closest to the ground truth did not result in the discovery of the most accurate and simple models. Similarly, we observed that simple and accurate agent-based models do not necessarily exhibit the best modularity characteristics. These findings suggest the need for new algorithms that can optimize across multiple dimensions (accuracy, simplicity, and modularity) when discovering agent-based models. Exploring such algorithms presents an interesting avenue for future work to advance insights produced by agent system mining. Finally, existing algorithms for discovering agent-based models from event data assume that the data includes attributes identifying the agent instances responsible for process activities, a requirement not imposed by traditional process discovery. It is clear that this reliance on additional information on agent instances partly ensures the added value of the discovered agent-based models. While we confirmed that, under

reasonable assumptions, most publicly available event logs contain this information, an interesting direction for future research is to explore whether this information can be inferred from standard event log attributes. This would significantly expand the applicability of our work. Another promising direction for future work is the development of new agent type identification techniques that construct MAS models specifically for agent-based simulation, enabling the generation of synthetic event data that accurately reflects the behavior of the original system.

Acknowledgments

Timotheus Kampik was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Camilo Alvarez, Eric Rojas, Michael Arias, Jorge Munoz-Gama, Marcos Sepúlveda, Valeria Herskovic, and Daniel Capurro. Discovering role interaction models in the emergency room using process mining. *J. Biomed. Informatics*, 78:60–77, 2018. URL https://doi.org/10.1016/j.jbi.2017.12.015.

[2] Rob H. Bemthuis and Sanja Lazarova-Molnar. Towards integrating process mining with agent-based modeling and simulation: State of the art and outlook. *Expert Syst. Appl.*, 281: 127571, 2025. URL https://doi.org/10.1016/j.eswa.2025.127571.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008. URL https://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

[4] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014. URL https://doi.org/10.1142/S0218843014400012.

[5] Andrea Burattin, Alessandro Sperduti, and Marco Veluscek. Business models enhancement through discovery of roles. In *CIDM*, pages 103–110, 2013. URL https://doi.org/10.1109/CIDM.2013.6597224.

[6] Cristina Cabanillas, Lars Ackermann, Stefan Schönig, Christian Sturm, and Jan Mendling. The ralph miner for automated discovery and verification of resource-aware process models. *Softw. Syst. Model.*, 19(6):1415–1441, 2020. URL https://doi.org/10.1007/s10270-020-00820-7.

[7] Berny Carrera and Jae-Yoon Jung. Constructing probabilistic process models based on hidden markov models for resource allocation. In *BPM Workshops*, volume 202 of *Lecture Notes in Business Information Processing*, pages 477–488, 2014. URL https://doi.org/10.1007/978-3-319-15895-2_41.

[8] Berny Carrera, JinSung Lee, and Jae-Yoon Jung. Discovering information diffusion processes based on hidden markov models for social network services. In *AP-BPM*, volume 219 of *Lecture Notes in Business Information Processing*, pages 170–182, 2015. URL https://doi.org/10.1007/978-3-319-19509-4_13.

[9] Valentino Crespi, Aram Galstyan, and Kristina Lerman. Top-down vs bottom-up method-

ologies in multi-agent system design. *Auton. Robots*, 24(3):303–313, 2008. URL https://doi.org/10.1007/s10514-007-9080-5.

[10] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005 (09):P09008, 2005. URL https://dx.doi.org/10.1088/1742-5468/2005/09/P09008.

[11] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, 2007. URL https://doi.org/10.1007/s10618-006-0061-7.

[12] Robert Dorfman. A formula for the gini coefficient. *The review of economics and statistics*, pages 146–149, 1979. URL https://doi.org/10.2307/1924845.

[13] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018. URL https://doi.org/10.1109/ACCESS.2018.2831228.

[14] Nicolas Dugué and Anthony Perez. Direction matters in complex networks: A theoretical and applied study for greedy modularity optimization. *Physica A: Statistical Mechanics and its Applications*, 603:127798, 2022. URL https://doi.org/10.1016/j.physa.2022.127798.

[15] Sam Fletcher and Md Zahidul Islam. Comparing sets of patterns with the jaccard index. *Australas. J. Inf. Syst.*, 22, 2018. URL https://doi.org/10.3127/ajis.v22i0.1538.

[16] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983. URL https://doi.org/10.1080/01621459.1983.10478008.

[17] Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In *BPM*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343, 2007. URL https://doi.org/10.1007/978-3-540-75183-0_24.

[18] Vagelio Kavakli and Pericles Loucopoulos. Goal-driven business process analysis application in electricity deregulation. *Inf. Syst.*, 24(3):187–207, 1999. URL https://doi.org/10.1016/S0306-4379(99)00015-0.

[19] Lukas Kirchdorfer, Robert Blümel, Timotheus Kampik, Han van der Aa, and Heiner Stuckenschmidt. Discovering multi-agent systems for resource-centric business process simulation. *Process Science*, 2(1):4, 2025. URL https://doi.org/10.1007/s44311-025-00009-5.

[20] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *BPM Workshops*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78, 2013. URL https://doi.org/10.1007/978-3-319-06257-0_6.

[21] Aaron F. McDaid, Derek Greene, and Neil J. Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *CoRR*, abs/1110.2515, 2011. URL https://doi.org/10.48550/arXiv.1110.2515.

[22] H. Van Dyke Parunak, Robert Savit, and Rick L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *MABS*, volume 1534 of *Lecture Notes in Computer Science*, pages 10–25, 1998. URL https://doi.org/10.1007/10692956_2.

[23] Artem Polyvyanyy, Matthias Weidlich, and Mathias Weske. Connectivity of workflow nets: the foundations of stepwise verification. *Acta Informatica*, 48(4):213–242, 2011. URL https://doi.org/10.1007/s00236-011-0137-8.

[24] Artem Polyvyanyy, Hanan Alkhammash, Claudio Di Ciccio, Luciano García-Bañuelos, Anna A. Kalenkova, Sander J. J. Leemans, Jan Mendling, Alistair Moffat, and Matthias Weidlich. Entropia: A family of entropy-based conformance checking measures for process mining. In *ICPM Doctoral Consortium / Tools*, volume 2703 of *CEUR Workshop Proceedings*, pages 39–42, 2020. URL https://ceur-ws.org/Vol-2703/paperTD6.pdf.

[25] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. Monotone precision and recall measures for comparing executions and specifi-

cations of dynamic systems. *ACM Trans. Softw. Eng. Methodol.*, 29(3):17:1–17:41, 2020. URL https://doi.org/10.1145/3387909.

[26] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. URL https://doi.org/10.1080/01621459.1971.10482356.

[27] Aubrey J. Rembert. Comprehensive workflow mining. In *ACM Southeast Regional Conference*, pages 222–227, 2006. URL https://doi.org/10.1145/1185448.1185498.

[28] Anne Rozinat, R. S. Mans, Minseok Song, and Wil M. P. van der Aalst. Discovering colored petri nets from event logs. *Int. J. Softw. Tools Technol. Transf.*, 10(1):57–74, 2008. URL https://doi.org/10.1007/s10009-007-0051-0.

[29] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. 2020. ISBN 9781292401133. URL http://aima.cs.berkeley.edu/.

[30] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the organisational perspective in agile business processes. In *BMMDS/EMMSAD*, volume 214 of *Lecture Notes in Business Information Processing*, pages 37–52, 2015. URL https://doi.org/10.1007/978-3-319-19237-6_3.

[31] Stefan Schönig, Cristina Cabanillas, Claudio Di Ciccio, Stefan Jablonski, and Jan Mendling. Mining resource assignments and teamwork compositions from process logs. *Softwaretechnik-Trends*, 36(4), 2016. URL https://dl.gi.de/handle/20.500.12116/40629.

[32] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. A framework for efficiently mining the organisational perspective of business processes. *Decis. Support Syst.*, 89:87–97, 2016. URL https://doi.org/10.1016/j.dss.2016.06.012.

[33] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. Self-organisation and emergence in MAS: an overview. *Informatica (Slovenia)*, 30(1):45–54, 2006. URL https://hal.science/hal-03811076.

[34] Qingtan Shen, Artem Polyvyanyy, Nir Lipovetzky, and Timotheus Kampik. Agent system event data: Concepts, dimensions, applications. In *ER*, volume 15238 of *Lecture Notes in Computer Science*, pages 56–72, 2024. URL https://doi.org/10.1007/978-3-031-75872-0_4.

[35] Minseok Song and Wil M. P. van der Aalst. Towards comprehensive support for organizational mining. *Decis. Support Syst.*, 46(1):300–317, 2008. URL https://doi.org/10.1016/j.dss.2008.07.002.

[36] Emilio Sulis and Kuldar Taveter. *Agent-Based Business Process Simulation - A Primer with Applications and Examples*. 2022. ISBN 978-3-030-98818-0. URL https://doi.org/10.1007/978-3-030-98816-6.

[37] Anja F. Syring, Niek Tax, and Wil M. P. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Trans. Petri Nets Other Model. Concurr.*, 14:192–221, 2019. URL https://doi.org/10.1007/978-3-662-60651-3_8.

[38] Niek Tax, Xixi Lu, Natalia Sidorova, Dirk Fahland, and Wil M. P. van der Aalst. The imprecisions of precision measures in process mining. *Inf. Process. Lett.*, 135:1–8, 2018. URL https://doi.org/10.1016/j.ipl.2018.01.013.

[39] Andrei Tour, Artem Polyvyanyy, and Anna A. Kalenkova. Agent system mining: Vision, benefits, and challenges. *IEEE Access*, 9:99480–99494, 2021. URL https://doi.org/10.1109/ACCESS.2021.3095464.

[40] Andrei Tour, Artem Polyvyanyy, Anna A. Kalenkova, and Arik Senderovich. Agent miner: An algorithm for discovering agent systems from event data. In *BPM*, volume 14159 of *Lecture Notes in Computer Science*, pages 284–302, 2023. URL https://doi.org/10.1007/978-3-031-41620-0_17.

[41] Wil M. P. van der Aalst. Verification of workflow nets. In *ICATPN*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426, 1997. URL https://doi.org/10.1007/

3-540-63139-9_48.

[42] Wil M. P. van der Aalst. Process mining: Overview and opportunities. *ACM Trans. Manag. Inf. Syst.*, 3(2):7:1–7:17, 2012. URL https://doi.org/10.1145/2229156.2229157.

[43] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. 2016. ISBN 978-3-662-49850-7. URL https://doi.org/10.1007/978-3-662-49851-4.

[44] Wil M. P. van der Aalst. Relating process models and event logs - 21 conformance propositions. In *ATAED@Petri Nets/ACSD*, volume 2115 of *CEUR Workshop Proceedings*, pages 56–74, 2018. URL https://ceur-ws.org/Vol-2115/ATAED2018-56-74.pdf.

[45] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004. URL https://doi.org/10.1109/TKDE.2004.47.

[46] Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects Comput.*, 23(3):333–363, 2011. URL https://doi.org/10.1007/s00165-010-0161-4.

[47] Anton JMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the HeuristicsMiner algorithm. 2006. URL https://pure.tue.nl/ws/portalfiles/portal/2388011/615595.pdf.

[48] Stephen Wolfram. Computation theory of cellular automata. *Communications in mathematical physics*, 96(1):15–57, 1984. URL https://doi.org/10.1007/BF01217347.

[49] Jing Yang, Chun Ouyang, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Yang Yu. *OrdinoR*: A framework for discovering, evaluating, and analyzing organizational models using event logs. *Decis. Support Syst.*, 158:113771, 2022. URL https://doi.org/10.1016/j.dss.2022.113771.