Mining Role-Based Behavioral Patterns From Event Data for Effective Process Simulation

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

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

Abstract. Roles and groups are fundamental components of agent systems, shaping both individual agent behavior and interactions among agents. Agent system mining enables the analysis of autonomous agents and their interaction patterns using event data generated by agent systems. However, existing agent system mining techniques primarily focus on agent-level information and often overlook role and group context of agents, potentially limiting their ability to capture agent behavioral patterns accurately. This limitation stems from the fact that explicit role and group information are usually absent from event data. To address this issue, we first formalize the concepts of roles, groups, and their relationships within agent systems. Building on this foundation, we propose Role Miner, a technique that infers roles and groups associated with agents for the events they perform. We further validate Role Miner using a novel Role Simulator that incorporates inferred role and group information as additional inputs. Experimental results show that our technique produces more accurate simulations, often outperforming state-of-the-art process simulation techniques.

Keywords: Agent system mining, role mining, agent-based simulation

1 Introduction

Process mining is a field that integrates concepts from data science and business process management (BPM) [1]. It aims to extract insights from event data to reveal system behavior patterns during process executions. Agent system mining (ASM) has emerged as a subarea of process mining [28], focusing on analyzing event logs generated by agent-based systems, which are composed of multiple autonomous agents that perform tasks and interact with others to achieve shared objectives. Such agents may represent software components, services, or human actors participating in a process. For instance, human workers can be considered as agents that collaborate to complete cases in organizational settings. ASM adopts an agent-oriented perspective, supporting agent-based process discovery (Agent Miner [29]) and simulation (Agent Simulator [15]). Compared with conventional process mining, ASM incorporates information about the agents who triggered each event to extract individual behavioral patterns and agent interaction structures. These extracted insights are then used to construct multi-agent system (MAS) models or to generate synthetic event logs. However, information about

roles and groups, which are important organizational concepts that can influence agent behavior [25], is typically not recorded in event data. As a result, existing agent-based discovery and simulation techniques often overlook this information, and the resulting models may fail to capture the organizational context that shapes agent behavior.

Both Agent Miner and Agent Simulator organize agents into agent types based on behavioral similarities. Agent Miner constructs behavioral and interaction models for each agent type, while Agent Simulator uses these agent type behavioral patterns to generate event traces. A key limitation of both approaches is that each agent is assigned to a single agent type for the entire event log, thereby assuming consistent behavior throughout the process. However, in agent-based systems, agents may play different roles across multiple groups within the same organization [10]. For example, an agent may act as a Ph.D. student in a research group while simultaneously serving as a tutor in a teaching group within the same institution. Consequently, an agent's behavior can vary depending on its role and group context throughout the process. Current ASM techniques do not capture such dynamic relationships among agents, roles, and groups. Role and organizational mining approaches exist within the BPM domain [27, 31]. However, they are not designed to infer roles as defined in agent-based systems, as they cannot model the assignment of different roles across groups for a single agent.

To address these limitations, we review organizational frameworks from agent-based systems to extract the concepts of agents, roles, groups, and their interrelationships. Building on these concepts, we propose Role Miner, which identifies the roles each agent plays within their groups during activity execution. Using event logs that include both conventional event attributes (i.e., case identifier, activity, and timestamp) and agent-related attributes (i.e., agent or resource instance), Role Miner provides a flexible representation that allows a single agent to play different roles across multiple groups throughout the process. Incorporating role and group information broadens the representational scope of agent system mining and increases the realism of simulated event logs. Furthermore, we propose Role Simulator, a novel agent-based simulation technique that integrates role and group information as additional inputs. Our evaluation shows that Role Simulator improves simulation accuracy compared with state-of-the-art process simulation techniques. The main contributions of this paper are:

- It reviews existing meta-models of agent-based systems and formalizes the definitions of roles and groups, which serve as the foundation for the design of Role Miner.
- It proposes the Role Miner technique, which infers role and group information for each agent that triggers an event in logs that lack such information.
- It introduces a novel agent-based simulation technique, Role Simulator, that incorporates role and group information as additional inputs for generating event logs.
- It evaluates the proposed simulation technique using real-world logs and compares its results with those from state-of-the-art process simulation techniques across multiple performance metrics, demonstrating that incorporating inferred role and group information can enhance simulation performance.

The paper is organized as follows. Section 2 reviews existing role mining techniques and process simulation techniques. Section 3 motivates Role Miner by comparing its mined organizational structure with that of existing organizational mining techniques. Section 4 presents Role Miner and Role Simulator in detail. Section 5 evaluates the

simulation results across multiple performance metrics and compares them with state-of-the-art process simulation techniques. Section 6 concludes the paper.

2 Related Work

This section reviews the foundational ontology of roles, role mining and organizational mining techniques, and automated business process simulation (BPS) approaches.

Masolo et al. provide an ontological characterization of social roles, highlighting key features such as their dynamic nature: a single entity can play multiple roles simultaneously [20]. Roles are also defined relative to multiple types of contexts, reflecting their contextual dependence [20]. Another work further emphasizes the relational aspect of roles, conceptualizing them as relations among multiple properties [19]. Guizzardi further formalizes the role concept using the Unified Foundational Ontology, proposing a grounded interpretation that reconciles competing views on role modeling [11].

Beyond the ontological perspective, roles have been studied in several application domains, with particular focus on BPM. Role-Based Access Control (RBAC) is one of the most widely adopted models for implementing advanced access control across diverse organizations [23]. In RBAC, roles serve as an intermediary between users and accessible resources. Building on the RBAC framework, role mining techniques have been developed to automatically infer roles from existing user-permission assignments [8]. Leitner et al. further demonstrate how role mining techniques can be applied to event logs to derive role models [17]. Within BPM, organizational mining [27] has emerged as a subfield of process mining that focuses on the discovery of organizational structures, roles, and social networks from event data. Comprehensive Workflow Mining [21] and Activity Matrix [27] are representative techniques that cluster agent instances into roles based on their activity preferences. More recently, the OrdinoR technique proposed mapping agent clusters into roles by jointly capturing similarities across multiple event attributes, including case identifiers, activities, and timestamps [31]. Additionally, Tour et al. proposed an agent type identification technique as part of Agent Miner, which grouped agents based on the similarity of their behavioral models [29].

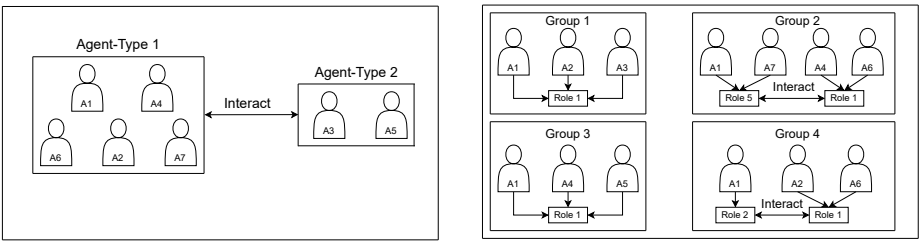
Role mining techniques based on the RBAC framework can infer roles from event data, allowing agents to hold multiple roles, which provides flexibility. However, a key limitation is that it does not specify the conditions under which an agent plays a particular role. As a result, these techniques cannot be directly applied to ASM, where precise knowledge of agent-role assignment in context is required to construct accurate models. The same limitation applies to OrdinoR [31]. Other organizational mining techniques [21, 27] can be applied to organize agents for ASM because they provide precise agent-role assignments. Other techniques, such as social network analysis [2] and behavior-based role mining by Jin et al. [13], offer mechanisms to group agents based on interaction patterns or behavioral similarity. However, they enforce a many-to-one agent-role assignment, where each agent is limited to a single role throughout the process. This rigid structure lacks flexibility and fails to capture the contextual variation in agent behavior. Additionally, Schönig et al. propose a framework for mining organizational perspective rules that requires pre-existing role and group information, limiting its applicability to conventional event logs [24].

Automated BPS has been addressed using a variety of approaches. Early work by Khodyrev et al. employs Petri net process models with execution-time analysis without considering resources for simulation [14], while Rozinat et al. use the colored Petri net models combined with discovered resource type behaviors and decision point analysis [22]. More recent techniques, such as Simod, generate process models using hyperparameter tuning, and construct resource pools by grouping resources with similar behaviors for simulation [5]. DSIM combines stochastic process models for activity execution with deep learning-based timestamp prediction, while assigning agents based on their activity execution profiles [6]. Finally, Agent Simulator introduces an agent-based simulation framework that extracts agent behaviors from historical logs, enabling agent-specific decision-making and interactions [15].

Overall, existing role mining techniques provide neither precise agent-role assignments nor flexible many-to-many agent-role assignments. Similarly, current BPS techniques do not consider agent roles within groups. These limitations motivate the development of a novel role mining technique that captures roles within group contexts, along with an agent-based process simulation technique using these contexts.

3 Motivating Example

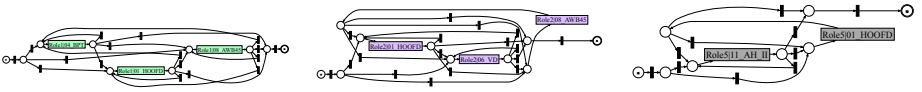
This section highlights the novelty of our technique by comparing the agent organizational structure obtained from a representative organizational mining method [29] with that produced by Role Miner. The comparison uses the real-world log *BPIC2015-1* from the Business Process Intelligence Challenge. A key characteristic of this log is that agents frequently collaborate within fixed groups across traces. This pattern suggests that the log reflects an agent-based system in which agents can play different roles depending on the group context in which they participate.



(a) Agent Miner (b) Role Miner

Fig. 1: Organizational structures discovered from the *BPIC2015-1* log.

First, when applying the Agent Miner clustering technique [29], the method constructs a directly-follows graph (DFG) for each agent based on its activity sequence. Agents are then clustered based on the similarity of their DFG-based behavioral models, assigning each agent to exactly one agent type. Figure 1a shows the organizational structure obtained using this technique on *BPIC2015-1*. For clarity, we show the structure using a subset of seven agents. These agents are grouped into two types, and all agents within the same type share an identical behavioral model. For example, *A1* belongs to *Agent-Type1* and always follows the behavioral model of this type throughout



(a) Role 1 (b) Role 2 (c) Role 5

Fig. 2: Behavioral models of the roles played by agent A1 throughout the process.

the process. Moreover, each agent type forms a single behavioral component, and only one interaction model is generated for the entire process, meaning that any inter-type agent interaction must follow this fixed interaction structure.

Second, we apply the Role Miner technique to the same log, producing the organizational structures shown in Fig. 1b. Again, focusing on the seven agents, they now form four groups. Each agent is assigned a specific role within each group, with roles labeled at the log level. For instance, *A1* plays *Role1* in *Group1*; *Role2* in *Group4*; and *Role5* in *Group2*. Each role is associated with its own behavioral model, allowing the same agent to exhibit different behaviors across groups. The behavioral models of these three roles, discovered using Inductive Miner [16], are shown in Fig. 2. Groups with similar sets of roles can be interpreted as pursuing similar business objectives and clustered into group types. Moreover, instead of modeling interactions between agent types, the structure now models interactions between roles within each group type, e.g., *Group1* and *Group3* share one role-interaction model, whereas *Group2* follows a different one.

Overall, the organizational structure generated by Role Miner is closely aligned with existing organizational frameworks in agent-based systems (see Section 4.1) and offers several advantages. First, it supports a flexible many-to-many agent–role assignment, allowing agents to play multiple roles across the process, while assigning roles based on their group information. Second, this structure defines multiple role–interaction models across different group types, enabling agent interaction patterns to vary based on the roles agents play and the groups they belong to. Finally, in process simulation, the framework supports role-based execution: each trace is first assigned to a group of agents with associated roles, and activities are predicted from role-specific behavioral patterns. For *BPIC2015-1*, using role-based simulation reduces the n-gram distance between simulated and test logs by 20% compared with state-of-the-art techniques, showing that incorporating roles and groups can yield more accurate simulations.

4 Approach

This section first reviews organizational frameworks from existing agent-based system meta-models. Building on these concepts, we propose Role Miner and detail its procedure for identifying roles and groups from event logs. Finally, we propose Role Simulator that uses these roles and groups to generate synthetic logs.

4.1 Agents, Roles, and Groups

Six studies propose frameworks for MASs, all of which incorporate the concept of roles and specify their functions within the organization. AALAADIN is an MAS meta-model that can be applied to construct organizations with hierarchical structures [9].

Table 1: Overview of major organizational framworks in agent-based systems; "✓" – supported; "❍" – not specified; "n–to–m" – a many-to-many relationship.

Framework	**Entity Relationships**			**Role Specifications**		
	Agent – Role	**Agent – Group**	**Role – Group**	**Within group**	**Role behavior**	**Role interaction**
AALAADIN [9]	n–to–m	n–to–m	n–to–m	✓	✓	✓
Gaia [30]	n–to–m	❍	❍	❍	✓	✓
MOISE [12]	n–to–m	n–to–m	n–to–m	✓	✓	✓
AGR [10]	n–to–m	n–to–m	n–to–m	✓	✓	✓
Framework by Boella et al. [3]	n–to–m	❍	n–to–m	✓	✓	✓
ASED [25]	n–to–m	n–to–m	n–to–m	✓	❍	✓

Gaia is a methodology for designing MAS that addresses both macro-level (societal) and micro-level (agent) aspects of the system [30]. MOISE is a meta-model that has gained some traction in agent-oriented programming [4, 12]. AGR is an MAS meta-model that addresses the limitations of MAS frameworks in software systems [10]. Boella et al. formalize roles from an ontological perspective, defining role interactions and specifying them within group contexts [3]. Finally, ASED [25] is the only meta-model that integrates both MAS organizational structures and business processes.

Table 1 summarizes the key aspects extracted from agent-based system meta-models into two components. The first component concerns entity relationships, which describe how agents, roles, and groups are related. Most meta-models adopt a many-to-many relationship for these entity pairs, i.e., a many-to-many agent–role relationship indicates that an agent may play multiple roles and that a role may be played by multiple agents. Exceptions include Gaia and the framework by Boella et al. [3], which focus on role-related relationships without specifying agent–group interactions. The second component concerns role specifications, which consist of three aspects. *Within group* indicates whether a role must be defined in the context of a group. All frameworks support this except Gaia, which focuses on role interactions. *Role behavior* indicates whether an agent's behavior is governed by the constraints and preferences associated with its role. All frameworks support this except ASED, which emphasizes the integrated participation of agents, roles, and groups as a collaborative *roleplay* in performing activities. *Role interaction* indicates whether roles can interact, e.g., perform task handovers. All reviewed frameworks support this aspect. These observations reveal six common conceptual aspects, forming the basis of our Role Miner technique with the following assumptions:

1. One agent can perform multiple roles; one role can be assigned to multiple agents.
2. One agent can participate in multiple groups; one group can include multiple agents.
3. One role can appear in multiple groups; one group can include multiple roles.
4. The role of an agent must be defined within the group in which the agent participates.
5. Within a group, an agent's behavior is determined by its role.
6. Roles can interact through handover tasks within a group.

In addition, we adopt the following assumptions:

Table 2: Agent activity matrices across three groups.

(a) Group G_1

	T_1	T_2	T_3	T_4
Ag_1	2	0	0	0
Ag_2	0	1	1	0
Ag_3	0	1	1	0

(b) Group G_2

	T_1	T_2	T_3	T_4
Ag_1	1	0	0	0
Ag_4	0	0	0	1
Ag_5	0	0	0	1

(c) Group G_3

	T_1	T_2	T_3	T_4
Ag_3	1	0	0	0
Ag_6	0	0	0	1
Ag_7	0	0	0	1

7. One group can trigger multiple traces; one trace is triggered by exactly one group.
8. One agent cannot play two different roles within the same group.
9. Groups that share the same set of roles belong to the same group type.

Regarding the group–trace relationship, each group is responsible for achieving its business objective, with agents in the group executing entire traces to fulfill it. Within a group, each agent assumes a fixed role to ensure consistent responsibilities. Group types are defined by role composition: groups with identical roles are assumed to pursue similar objectives and are clustered under the same type. We note that the last three assumptions are not intended as universal properties of roles; rather, they provide structural constraints that facilitate the design of Role Miner.

4.2 Role Miner

The Role Miner operates in four steps, which we illustrate using a sample event log as input. The log consists of four traces and involves four activities and seven agents, denoted as $L = \{C_1, C_2, C_3, C_4\}$. Each trace comprises an ordered sequence of events, where each event is characterized by an event identifier, case identifier, activity, timestamp, and agent instance. Since the ordering of events within each trace is determined by timestamps, we present only the activity T and the agent instance Ag for each event for clarity: $C_1 = \langle (Ag_1, T_1), (Ag_2, T_2), (Ag_3, T_3) \rangle$; $C_2 = \langle (Ag_2, T_3), (Ag_1, T_1), (Ag_3, T_2) \rangle$; $C_3 = \langle (Ag_1, T_1), (Ag_4, T_4), (Ag_5, T_4) \rangle$; $C_4 = \langle (Ag_3, T_1), (Ag_6, T_4), (Ag_7, T_4) \rangle$.

Step 1 – clustering agents into groups. We begin by identifying, for each trace, the set of agents involved in generating it. For example, the agent set generating trace C_1 is $\{Ag_1, Ag_2, Ag_3\}$. We then collect the agent sets for all traces and count the number of traces generated by each agent set. For the log L, the result is $L_f = \{\{Ag_1, Ag_2, Ag_3\}^2, \{Ag_1, Ag_4, Ag_5\}^1, \{Ag_3, Ag_6, Ag_7\}^1\}$, where the superscript denotes the number of traces generated by each agent set. Next, we apply a threshold $t_1 \in [0, 1]$, representing the minimum percentage of traces in which a group must appear, and remove all groups whose frequencies fall below $t_1 \times |L|$. This step allows us to retain only those groups that occur sufficiently often to reflect stable organizational patterns. For instance, with $t_1 = 0.2$, the minimum required frequency is $4 \times 0.2 = 0.8$; thus, none of the groups in L_f are removed. If users aim to capture all group information, they can set $t_1 = 0$, which retains every group in the log. Even under this setting, the number of resulting group types does not necessarily increase, suggesting that the overall framework does not grow in complexity. After applying the threshold, the group–trace associations are as follows: $G_1 = \{Ag_1, Ag_2, Ag_3\}$ executing $\{C_1, C_2\}$; $G_2 = \{Ag_1, Ag_4, Ag_5\}$ executing $\{C_3\}$, and $G_3 = \{Ag_3, Ag_6, Ag_7\}$ executing $\{C_4\}$.

Table 3: Activity matrix for group-level roles.

	T_1	T_2	T_3	T_4
$G_1\|R_1$	2	0	0	0
$G_1\|R_2$	0	2	2	0
$G_2\|R_1$	1	0	0	0
$G_2\|R_2$	0	0	0	2
$G_3\|R_1$	1	0	0	0
$G_3\|R_2$	0	0	0	2

Table 4: Final Role Miner output.

Group type	Group	Role	Agent
GT_1	G_1	R_1	Ag_1
GT_1	G_1	R_2	Ag_2&Ag_3
GT_2	G_2	R_1	Ag_1
GT_2	G_2	R_3	Ag_4&Ag_5
GT_2	G_3	R_1	Ag_3
GT_2	G_3	R_3	Ag_6&Ag_7

Step 2 – mining roles within each group. In this step, we focus on mining roles within each group based on the similarity of activity preferences among agents. The identified roles are strictly defined at the group level. To perform clustering, we adopt the Activity Matrix approach [27]. For each group, we first construct an activity matrix, where the rows correspond to agents in the group and the columns correspond to all activities observed across the entire log. Table 2 shows the activity matrices for the three groups. Each cell in the matrix records the frequency with which a given agent triggers a particular activity across all traces executed by the corresponding group. For instance, agent Ag_1 performs activity T_1 twice across the two traces of G_1, yielding $(Ag_1, T_1) = 2$ in the matrix for G_1. Next, we compute the Pearson correlation coefficient for each pair of agents within the matrix. A higher correlation indicates greater similarity in activity behavior. To ensure meaningful clustering, a threshold $t_2 \in [0, 1]$ is applied to filter out agents whose correlation falls below the minimum acceptable level. For example, the correlation between $\vec{Ag_2} = [0, 1, 1, 0]$ and $\vec{Ag_3} = [0, 1, 1, 0]$ is 1; if we set $t_2 = 0.9$, these agents are clustered in the same role. Conversely, Ag_1 shows low correlation with Ag_2 and Ag_3, forming a distinct role. Finally, we construct a correlation graph in which agents are represented as nodes, and edges connect pairs of agents whose correlation exceeds the threshold t_2. Each connected agent component in this graph corresponds to a distinct role. The resulting roles are: for G_1, $\{Ag_1\}$ plays role $G_1|R_1$, and $\{Ag_2, Ag_3\}$ plays $G_1|R_2$; for G_2, $\{Ag_1\}$ plays role $G_2|R_1$, and $\{Ag_4, Ag_5\}$ plays $G_2|R_2$; for G_1, $\{Ag_3\}$ plays role $G_3|R_1$, and $\{Ag_6, Ag_7\}$ plays $G_3|R_2$. Note that roles are group-specific, i.e., $G_1|R_1$ and $G_2|R_1$ are distinct roles because they are defined within different group contexts.

Step 3 – clustering roles at log level. In this step, we cluster the group-level roles obtained in *Step 2* into log-level roles using the Activity Matrix approach [27]. We begin by constructing an activity matrix in which each row corresponds to a group-level role, and the columns represent all activities. Each cell records the total frequency with which the agents assigned to a given role trigger a particular activity across all traces executed by the corresponding group. Table 3 shows the resulting matrix for the six group-level roles. For instance, role $G_1|R_2$ triggers T_2 twice because agents Ag_2 and Ag_3 both play this role and each triggers T_2 once. Next, we compute the correlation for every pair of group-level roles. A threshold $t_3 \in [0, 1]$ is then applied to construct a correlation graph in which nodes represent group-level roles and edges connect pairs whose correlation exceeds t_3. Each connected component of this graph corresponds to a distinct log-level role. With $t_3 = 0.9$, the resulting log-level roles are $R_1 = \{G_1|R_1, G_2|R_1, G_3|R_1\}$, $R_2 = \{G_1|R_2\}$, and $R_3 = \{G_2|R_2, G_3|R_2\}$. These log-level roles represent behavioral patterns that hold across groups and thus can be interpreted as organizational-level roles.

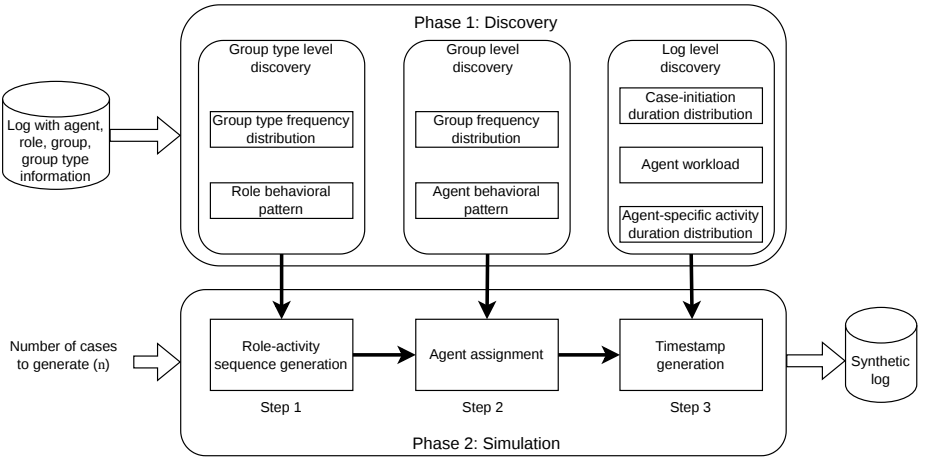


Fig. 3: Role Simulator procedure.

Step 4 – identifying group types. Finally, we identify group types by analyzing the log-level roles present in each group. The underlying assumption is that groups involving similar sets of organizational-level roles pursue similar business objectives and thus belong to the same group type. For each group, we first derive the set of log-level roles performed by its agents. Groups with identical role sets are then clustered into the same group type, e.g., both G_2 and G_3 have the role set $\{R_1, R_3\}$ and are thus assigned to the same group type. In contrast, G_1 has the role set $\{R_1, R_2\}$, which differs from that of G_2 and G_3, and therefore forms a separate group type. Table 4 shows the resulting organizational framework produced by Role Miner. The framework captures the flexibility of agent behavior within the organization, e.g., agent Ag_3 plays role R_2 in G_1 but plays role R_1 in G_3. Moreover, for every event in the log, we can assign its role unambiguously by considering the performing agent and the group responsible for the corresponding trace. This yields an enriched event log annotated with group types, groups, and roles, which can subsequently serve as input for downstream agent-system mining techniques.

4.3 Role Simulator

Role Simulator is operated on an enriched event log that extends the conventional log with additional organizational attributes, including agent, role, group, and group type. For illustration, we use the sample log produced by Role Miner as input to demonstrate each step of Role Simulator. An additional parameter required by the simulator is the number of cases to generate. Figure 3 presents an overview of the procedure, which consists of a discovery phase that extracts organizational and behavioral knowledge from the input log, followed by a simulation phase that generates synthetic logs using this knowledge. In the following paragraphs, we describe these three simulation steps and explain how the corresponding knowledge is discovered and applied.

Step 1 – role-activity sequence generation. When a new case is initiated, we first generate its sequence of role-activity pairs, where each pair corresponds to an event. To enable this generation, two types of information must be extracted from the log.

First, we compute frequency distributions of group types. This yields the probability that a case is executed by each group type, denoted as $P(GT_i|C)$, e.g., in the sample log, two out of four cases are executed by GT_2, then $P(GT_2|C) = 2/4$. Second, for each group type, we derive the conditional probabilities of the next role-activity pair given a preceding sequence. To do this, we collect all cases associated with a particular group type and extract their sequences of role-activity pairs. From these sequences, we compile frequency counts for all prefixes of any length, and then compute the probability of each possible next pair. Start (S) and end (E) markers are also included in the probability calculation. For example, two cases from GT_1 with roles-activity pairs are $C_1 = \langle (R_1, T_1), (R_2, T_2), (R_2, T_3) \rangle$ and $C_2 = \langle (R_2, T_3), (R_1, T_1), (R_2, T_2) \rangle$. If the preceding sequence is $\langle (R_2, T_2) \rangle$, the next element is (R_2, T_3) once, and end (E) once; thus, $P((R_2, T_3)|\langle (R_2, T_2) \rangle, GT_1) = 1/2$, and $P(E|\langle (R_2, T_2) \rangle, GT_1) = 1/2$. Longer preceding sequences are handled similarly, i.e., $P((R_2, T_3)|\langle (R_1, T_1), (R_2, T_2) \rangle, GT_1) = 1$. During simulation, when a new case C_j arrives, we first assign it to a group type according to $P(GT_i|C_j)$. Given the assigned group type GT_j, the first role-activity pair is sampled from $P((R_i, T_i)|\langle S \rangle, GT_j)$. Suppose the sampled pair is (R_j, T_j), the next pair is predicted using $P((R_i, T_i)|\langle S, (R_j, T_j) \rangle, GT_j)$. If no probability is defined for a particular preceding sequence, we iteratively shorten the sequence (removing the earliest element each time) until a valid probability distribution is obtained. Finally, when the end symbol E is generated, the sequence terminates. Its assigned group type and the resulting role-activity sequence is then used in the subsequent steps of the simulation.

Step 2 – agent assignment. Once the role–activity sequence for a case is generated, the next step is to assign an agent to each event. This requires extracting two types of information from the log. First, we compute the frequency distribution of groups within each group type. For a given group type GT_i, we compute the proportion of cases executed by each group belonging to that type. E.g., under GT_2, groups G_2 and G_3 each execute one case; thus $P(G_2|GT_2) = P(G_3|GT_2) = 1/2$. Second, for each group, we derive the conditional probabilities of the next agent-activity pair given a preceding sequence. To do this, we collect all cases associated with a given group and extract their sequences of agent-activity pairs. For every preceding sequence (including the Start symbol S), we count the occurrences of all possible next agent-activity pairs and compute their proportions. E.g., in group G_1, considering the prefix $\langle (Ag_1, T_1) \rangle$, the next agent-activity pair occurs once as (Ag_2, T_2) and once as (Ag_3, T_2); thus, $P((Ag_2, T_2)|\langle (Ag_1, T_1) \rangle, G_1) = 1/2$. Longer preceding sequences are handled similarly. To assign an agent, we condition both the preceding agent-activity sequence and the next role-activity pair; e.g., given prefix $\langle (Ag_1, T_1) \rangle$ in group G_1 and the next role-activity pair (R_2, T_2), the probability is $P(Ag_j|\langle (Ag_1, T_1) \rangle, G_1, R_2, T_2) = P((Ag_j, T_2)|\langle (Ag_1, T_1) \rangle, G_1) / \sum_{Ag_i \in R_2} P((Ag_i, T_2)|\langle (Ag_1, T_1) \rangle, G_1)$ for assigning agent Ag_j. This normalizes the likelihoods of all agents capable of performing role R_2 and triggers T_2, ensuring that the assigned agent is consistent with the required role. During the agent assignment, we first assign the case to a group G_j according to the distribution $P(G_i|GT_j)$ of its assigned group type GT_j. If the first role-activity pair of the case is (R_j, T_j), the agent for the first event is sampled from $P(Ag_i|\langle S \rangle, G_j, R_j, T_j)$. If this yields Ag_j, and the next role-activity pair is (R_k, T_k), the agent for the next event is sampled from $P(Ag_i|\langle S, (Ag_j, T_k) \rangle, G_j, R_k, T_k)$. This process continues until all events in the sequence have been assigned an agent.

Step 3 – timestamp generation. Once agents have been assigned to all events, the final step is to predict start and end timestamps for every event. To achieve this, three types of temporal knowledge are discovered from the input log. First, we extract the case-initiation duration distribution, obtained from the inter-arrival times between consecutive case start timestamps. Outliers are removed, and a probabilistic model (i.e., the gamma distribution) is fitted to the cleaned durations. This distribution is then used to predict the arrival time of each new case, thereby determining its start timestamp. Second, we discover agent availability patterns, which capture the working-time intervals during which agent perform activities. This knowledge is extracted at the log level, because agent availability is considered an organizational characteristic rather than a role- or group-specific one. For each agent, event timestamps are aggregated by week to identify time intervals containing at least one event execution. Intervals with observed executions are labeled as active, while intervals without executions are considered inactive. During simulation, if an activity is scheduled to start while its assigned agent is inactive, its execution is postponed until the next available interval. Third, we learn the agent-specific activity duration distributions. For each agent–activity pair, we compute the observed durations (end time minus start time) and fit a probability distribution to these durations. Once the start time of an activity is determined, a sampled duration from this distribution is added to estimate the corresponding end time. After assigning timestamps to all events, the complete synthetic log is generated.

5 Evaluation

This section evaluates Role Simulator (RoleSim) using event logs enriched with organizational information inferred by Role Miner. We begin by describing the event logs used in the evaluation and outlining the experimental setup, and then compare logs generated by RoleSim with those from two state-of-the-art process simulation techniques, Simod [5] and Agent Simulator [15] (AgentSim).

5.1 Datasets and Experimental Setup

We use a total of 14 publicly available event logs in our experiments, including 4 synthetic logs (syn) and 10 real-world logs (real) commonly used in the agent system mining literature [15, 26, 29]. The *original attributes* columns in Table 5 summarize the basic characteristics of each log. We compare our RoleSim against two state-of-the-art business process simulation techniques, both of which place strong emphasis on the resource perspective, including resource capabilities and behavioral patterns. The first technique is the data-driven process simulation approach [18], referred to as Simod which extends the original Simod framework [5] with the Prosimos simulator. This integration incorporates resource availability considerations and was shown to outperform the original Simod. The second technique is AgentSim [15], a recent resource-centric framework that models agent-specific decision-making and interactions, demonstrating competitive performance relative to other simulation approaches. While Simod emphasizes process model structures, AgentSim focuses on agent behavior probabilities,

Table 5: Description of log properties (with discovered organizational attributes).

		Original attributes				Organizational attributes (90% traces)			Simulation metric
Event log	**Type**	**#Traces**	**#Events**	**#Activities**	**#Agents**	**#Roles**	**#Groups**	**#Group-types**	**RoleSIM group-overlap**
Loan Application	syn	1,000	7,492	12	19	5	49	2	0
CVS	syn	10,000	103,906	15	8	5	11	2	75.3%
Confidential 1000	syn	1,000	38,160	42	26	18	37	23	0.1%
Confidential 2000	syn	2,000	77,418	42	26	18	29	19	0.3%
ACR	real	954	6,870	18	432	9	18	9	0
BPIC2012W	real	8,616	59,302	6	56	3	23	3	21.6%
BPIC2014	real	46,616	466,737	39	242	16	15	15	24.3%
BPIC2015-1	real	1,199	52,217	20	23	8	22	12	19.4%
BPIC2015-2	real	832	44,354	26	11	5	19	7	21.4%
BPIC2015-3	real	1,409	59,681	23	14	3	29	4	30.9%
BPIC2015-4	real	1,053	47,293	26	10	3	11	4	15.7%
BPIC2015-5	real	1,156	59,083	24	22	8	27	12	14.5%
BPIC2017W	real	30,276	240,854	8	136	1	18	1	1.1%
BPIC2020	real	7,065	86,581	21	8	8	8	8	16.5%

providing two complementary perspectives for comparison. Role Miner and Role Simulator (RoleSim) code is publicly available.[3]

For the experiments, we first partition each event log into a train log comprising 90% of the traces and a test log comprising the remaining 10%. We then apply the Role Miner technique to the train logs to infer the roles, groups and group types for agents triggering each event, using thresholds $t_1 = 0.01$, $t_2 = 0.95$, and $t_3 = 0.95$. The resulting inferred attributes are summarized in the *organizational attributes* columns in Table 5. The enriched train logs, possessing both original and inferred attributes, are then used as input to RoleSim, which is configured to generate the same number of traces as the test log. We run RoleSim ten times, generating ten simulated logs. We then input the same train log into Simod and AgentSim, generating ten logs for each technique.

To compare the different simulation approaches, we use a set of recently proposed metrics that evaluate performance from control-flow, temporal, and congestion perspectives [7]. Control-flow perspective is measured using the N-Gram Distance (NGD), which quantifies differences in the frequency of n-grams ($n = 3$) observed in the logs. Temporal behavior is measured using three metrics that capture distinct aspects of event timing: Absolute Event Distribution (AED), Circadian Event Distribution (CED), and Relative Event Distribution (RED). The ability to reproduce congestion is evaluated using the Cycle Time Distribution (CTD). All metrics are computed exactly as in AgentSim, ensuring a consistent basis for comparison. For all metrics, distances are computed between each of the ten simulated logs and the corresponding test log, and the mean distance across the ten simulations is reported. Lower mean values indicate a closer match to the test log and, thus, better simulation performance. For our trace-based technique, no warm-up period is required, and for both compared techniques, we use their default parameter settings.

5.2 Results

Table 6 reports the simulation results for the 14 logs, showing the average values of each metric across the 10 generated simulation logs. The best value is highlighted in bold, and the second-best is underlined. No single technique performs best across all

[3] https://github.com/QINGTANS/Role-Miner-and-Simulator

Table 6: Comparison of simulation techniques.

Event log	Method	NGD	AED	CED	RED	CTD	Event log	Method	NGD	AED	CED	RED	CTD
LoanApp	**Simod**	0.13	14.67	0.38	6.33	9.50	**CVS**	**Simod**	0.47	**276.70**	0.85	**40.24**	**42.21**
LoanApp	**AgentSim**	**0.06**	**3.48**	**0.23**	**2.41**	**3.51**	**CVS**	**AgentSim**	0.06	407.20	**0.34**	168.23	275.26
LoanApp	**RoleSim**	0.46	86.83	4.86	5.23	7.72	**CVS**	**RoleSim**	**0.05**	592.70	5.43	148.87	245.24
Conf1000	**Simod**	0.27	**54.98**	**2.42**	208.24	327.34	**Conf2000**	**Simod**	**0.24**	**160.21**	3.47	389.28	598.99
Conf1000	**AgentSim**	**0.25**	165.20	2.90	44.68	72.56	**Conf2000**	**AgentSim**	0.25	196.70	3.72	28.25	46.25
Conf1000	**RoleSim**	0.38	229.61	2.49	**2.28**	**3.25**	**Conf2000**	**RoleSim**	0.32	361.99	**2.40**	**4.61**	**8.52**
ACR	**Simod**	0.42	**185.36**	**3.24**	113.41	327.89	**BPIC2012W**	**Simod**	0.49	**209.69**	2.00	**36.17**	**39.61**
ACR	**AgentSim**	0.38	417.23	5.45	22.54	66.28	**BPIC2012W**	**AgentSim**	**0.13**	292.74	**1.82**	129.04	155.15
ACR	**RoleSim**	**0.23**	283.28	3.54	**19.42**	**45.10**	**BPIC2012W**	**RoleSim**	0.70	263.84	3.06	128.87	154.96
BPIC2014	**Simod**	n/a	n/a	n/a	n/a	n/a	**BPIC2015-1**	**Simod**	0.44	21753.46	**3.37**	20631.27	21550.47
BPIC2014	**AgentSim**	0.36	39.43	4.69	55.23	59.90	**BPIC2015-1**	**AgentSim**	0.26	2104.29	13.98	678.50	1021.54
BPIC2014	**RoleSim**	**0.23**	**23.69**	**2.59**	**3.31**	**4.65**	**BPIC2015-1**	**RoleSim**	**0.20**	**1532.64**	4.27	**623.01**	**937.38**
BPIC2015-2	**Simod**	0.78	85933.24	**5.33**	84763.32	83043.64	**BPIC2015-3**	**Simod**	0.65	10274.18	**4.10**	9732.72	13821.19
BPIC2015-2	**AgentSim**	0.36	1785.28	6.79	797.46	1130.18	**BPIC2015-3**	**AgentSim**	0.26	2290.44	17.10	646.19	828.42
BPIC2015-2	**RoleSim**	**0.33**	**1437.42**	5.88	**730.27**	**1021.72**	**BPIC2015-3**	**RoleSim**	**0.25**	**1356.77**	6.11	**580.35**	**727.77**
BPIC2015-4	**Simod**	0.98	357429.14	**5.44**	356414.04	643344.48	**BPIC2015-5**	**Simod**	0.93	200854.48	6.69	200021.65	146764.93
BPIC2015-4	**AgentSim**	**0.35**	2214.63	10.63	407.10	665.89	**BPIC2015-5**	**AgentSim**	0.45	2286.55	6.99	554.35	824.42
BPIC2015-4	**RoleSim**	0.44	**1159.33**	6.26	**336.19**	**589.24**	**BPIC2015-5**	**RoleSim**	**0.36**	**1963.85**	**6.00**	**501.25**	**689.38**
BPIC2017W	**Simod**	0.35	**332.57**	3.49	**22.02**	**89.43**	**BPIC2020**	**Simod**	0.43	**402.57**	**2.65**	**420.61**	**299.48**
BPIC2017W	**AgentSim**	**0.20**	613.58	**1.47**	204.91	268.13	**BPIC2020**	**AgentSim**	**0.38**	595.71	8.04	560.40	1835.55
BPIC2017W	**RoleSim**	0.62	803.82	2.88	205.04	267.30	**BPIC2020**	**RoleSim**	0.40	749.25	2.88	552.92	1813.31

logs. RoleSim achieves the strongest overall performance, with the highest number of wins across the NGD, RED, and CTD metrics, and performs comparably to Simod on AED. From a control-flow perspective, RoleSim performs best on NGD, winning 7 logs, compared with AgentSim (6 wins) and Simod (1 win). RoleSim outperforms the process-model-based method (Simod), demonstrating its ability to generate accurate activity sequences by incorporating groups and roles as organizational structures to constrain agent behavior. It can also outperform the resource-oriented approach, which does not account for the organizational structures typically present in real-world logs.

From a temporal perspective, RoleSim achieves the best results on RED, indicating a stronger ability to reproduce the relative timing patterns. It also performs comparably to Simod on AED, though it falls behind Simod on CED. This performance stems from the simulation design: each trace is assigned to a group of agents who collaboratively execute all events in the trace. Agents are not selected from a global resource pool or driven solely by independent probabilistic models; instead, they act within a shared group context. This structure increases temporal coherence within a trace, as the same agents remain responsible for activities throughout the trace. Although this can introduce some inflexibility, e.g., waiting for an inactive agent rather than reassigning tasks to agents outside the group. From a congestion perspective, RoleSim outperforms others on CTD, suggesting more accurate reproduction of cycle times. This is because each group employs the same fixed agents across traces, preserving their workload and activity-duration patterns and producing more consistent cycle-time behavior.

To strengthen the comparison, we introduce the group-overlap metric, which measures the average proportion of traces executed by exactly the same agent groups across the 10 simulated logs relative to the test log. The last column of Table 5 reports the group-overlap values for RoleSim, while both Simod and AgentSim show zero group overlap across all 14 logs, indicating that they fail to reproduce the agent groups collaborating to execute traces. Notably, for the *BPIC2014* and the five *BPIC2015* logs,

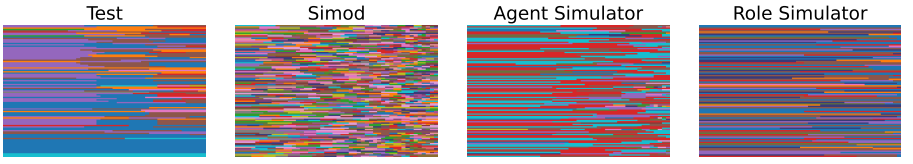


Fig. 4: Trace–agent map of *BPIC2015-1* test and simulated logs: rows are traces, columns from left to right show consecutive events, colors indicate agents.

RoleSim achieves higher group overlap values and simultaneously outperforms the other techniques on most metrics. This shows a strong link between preserving agent-group structures and improving simulation accuracy, highlighting the value of modeling groups and their internal role interactions when reproducing real-case behavior. However, for *CVS*, the highest group-overlap does not correspond to superior simulation performance, suggesting the behavioral patterns may not be strictly role-based. For *ACR*, behaviors remain largely consistent with roles despite incorrectly inferred groups, allowing RoleSim to still generate reasonable simulations. Overall, these observations suggest that the techniques perform well when agents follow role-based behaviors, particularly when group collaboration influences trace execution.

To evaluate simulation accuracy from the agent perspective, Fig. 4 illustrates agent interaction for the test and simulated logs of *BPIC2015-1*. Each row represents a trace, shown from left to right as consecutive normalized number of events, with each color representing a specific agent. For Simod, agents are assigned randomly, resulting in more task handovers and more agents per trace than in the test log. AgentSim reduces handovers but overrepresents certain agents (e.g., light blue and red) compared to the test log. In contrast, RoleSim closely matches the number of agents and handovers per trace, and the agent color frequencies align the most closely with the test log. This demonstrates that RoleSim can generate more accurate agent interaction patterns by modeling agents within groups and roles. Based on these patterns, RoleSim achieves high-quality results across all metrics from the three perspectives, except for CED, highlighting the value of incorporating organizational structures into the simulation. Finally, RoleSim is more efficient: for the largest log *BPIC2014*, it runs with RoleMiner in approximately 3 minutes on a 16 GB RAM, 10-core M2 Pro (3.5 GHz CPU), which is roughly 10 times faster than AgentSim, while Simod runs out of memory.

Regarding experimental limitations, the first is the lack of ground-truth roles in the logs, which prevents direct validation of roles mined by RoleMiner. Second, we assume that all logs originate from agent-based systems with roles and groups, while the RoleSim group-overlap metric suggests that many real-world logs exhibit higher group-overlap, partially supporting this assumption.

The Role Miner and RoleSim also have limitations. First, Role Miner relies on manually specified thresholds. While certain threshold values yield clear organizational structures from some logs, they may be inappropriate for logs that contain heterogeneous behavioral domains. Second, the filtering step removes traces executed by infrequent groups, which may hinder the discovery of important behavioral patterns. Third, the current design assumes that each trace is executed by a single group, whereas in practice, multiple groups may collaborate within the same trace. Future improvements include automatically selecting thresholds based on patterns learned from diverse logs,

replacing trace filtering with clustering of infrequent groups into similar group types, and extending the framework to support multi-group collaboration within a single trace.

Turning to the RoleSim limitations, first, although RoleSim uses role and group information from Role Miner, it generates activity sequences at the group level rather than modeling interactions between individual roles. This preserves group-level behavior but omits detailed role-to-role dynamics shown in Fig. 1b. Future work could incorporate explicit role interactions. Second, discovering agent interaction patterns separately for each group can become computationally complex when many groups exist. Third, agents in the current simulation are not fully autonomous: roles and activities are generated before assigning agents, which limits the exploitation of the agent–role relationship. Fourth, the simulator does not account for resource contention or potential conflicts when an agent is assigned to multiple roles simultaneously, representing a potential area for improvement. Future improvements include clustering groups with similar behavioral characteristics so they can share interaction patterns, thereby reducing complexity. Moreover, the simulator could be enhanced by increasing agents' autonomy, with roles and groups treated as intermediate layers from which agents can select. By learning the models of how agents form groups and switch roles, the simulator could generate events from the contextual behavioral patterns associated with an agent's assigned role within a group.

Despite these limitations, the proposed approach has practical relevance. It provides a data-driven way to infer role and group information from event logs without predefined organizational information. This can help organizations understand how agent responsibilities and collaborations perform in practice, particularly where roles are dynamic. The inferred information can support organizational analysis tasks such as workforce planning and resource allocation. Moreover, by incorporating this information into agent-based process simulation, organizations can evaluate potential changes in role assignments or collaboration patterns before applying them in practice. In future work, the simulation framework could support what-if analysis, allowing organizations to explore alternative agent-role and agent-group assignments, adjust role-specific transition probabilities and interaction structures, and simulate the resulting process executions to assess their potential impact.

6 Conclusion

This paper first reviews the concepts of roles and groups from agent-based systems. Building on these concepts, we propose Role Miner, which infers roles and groups to construct a flexible organizational structure of agents. We then propose Role Simulator, which uses the inferred roles and groups to perform process simulation. The simulator demonstrates competitive performance and greater efficiency compared to state-of-the-art process simulation techniques. Future work could enhance agent-based simulation by clustering groups to share interaction patterns and improving agent autonomy through discovered probabilistic models for group formation and role switching. Overall, grounded in the reviewed role and group concepts, Role Miner generates valuable outputs for understanding organizational structures and supporting downstream tasks, while Role Simulator demonstrates the practical benefits of the inferred roles and groups with effective performance.

References

[1] van der Aalst, W.M.P.: Process Mining—Data Science in Action. Springer, 2nd edn. (2016)
[2] van der Aalst, W.M.P., Reijers, H.A., Song, M.: Discovering Social Networks from Event Logs. Comput. Support. Cooperative Work. **14**, 549–593 (2005)
[3] Boella, G., van der Torre, L.W.N.: The Ontological Properties of Social Roles in Multi-agent Systems: Definitional Dependence, Powers and Roles Playing Roles. Artif. Intell. Law **15**, 201–221 (2007)
[4] Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent Oriented Programming with JaCaMo. Science of Computer Programming **78**, 747–761 (2013)
[5] Camargo, M., Dumas, M., González, O.: Automated Discovery of Business Process Simulation Models from Event Logs. Decis. Support Syst. **134**, 113284 (2020)
[6] Camargo, M., Dumas, M., Rojas, O.G.: Learning Accurate Business Process Simulation Models from Event Logs via Automated Process Discovery and Deep Learning. In: CAiSE, pp. 55–71 (2022)
[7] Chapela-Campa, D., Benchekroun, I., Baron, O., Dumas, M., Krass, D., Senderovich, A.: Can I Trust My Simulation Model? Measuring the Quality of Business Process Simulation Models. In: BPM, pp. 20–37 (2023)
[8] Ene, A., Horne, W.G., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast Exact and Heuristic Methods for Role Minimization Problems. In: SACMAT, pp. 1–10 (2008)
[9] Ferber, J., Gutknecht, O.: A Meta-model for the Analysis and Design of Organizations in Multi-agent Systems. In: ICMAS, pp. 128–135 (1998)
[10] Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-agent Systems. In: AOSE, pp. 214–230 (2003)
[11] Guizzardi, G.: Agent Roles, Qua Individuals and the Counting Problem. In: SELMAS, pp. 143–160 (2005)
[12] Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: MOISE: An Organizational Model for Multi-agent Systems. In: IBERAMIA-SBIA, pp. 156–165 (2000)
[13] Jin, T., Wang, J., Wen, L.: Organizational Modeling from Event Logs. In: GCC, pp. 670–675 (2007)
[14] Khodyrev, I., Popova, S.: Discrete Modeling and Simulation of Business Processes Using Event Logs. In: ICCS, pp. 322–331 (2014)
[15] Kirchdorfer, L., Blümel, R., Kampik, T., van der Aa, H., Stuckenschmidt, H.: Discovering Multi-agent Systems for Resource-centric Business Process Simulation. Process Science **2**, 4 (2025)
[16] Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-structured Process Models from Event Logs Containing Infrequent Behaviour. In: BPM Workshops, pp. 66–78 (2013)
[17] Leitner, M., Baumgrass, A., Schefer-Wenzl, S., Rinderle-Ma, S., Strembeck, M.: A Case Study on the Suitability of Process Mining to Produce Current-state RBAC Models. In: BPM Workshops, pp. 719–724 (2012)
[18] López-Pintado, O., Dumas, M.: Business Process Simulation with Differentiated Resources: Does it Make a Difference? In: BPM, pp. 361–378 (2022)
[19] Masolo, C., Guizzardi, G., Vieu, L., Bottazzi, E., Ferrario, R., et al.: Relational Roles and Qua-individuals. In: AAAI Fall Symposium on Roles, an interdisciplinary perspective, pp. 103–112 (2005)
[20] Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social Roles and their Descriptions. In: KR, pp. 267–277 (2004)
[21] Rembert, A.J.: Comprehensive Workflow Mining. In: ACM Southeast Regional Conference, pp. 222–227 (2006)

[22] Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering Simulation Models. Inf. Syst. **34**, 305–327 (2009)
[23] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based Access Control Models. Computer **29**, 38–47 (1996)
[24] Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: A Framework for Efficiently Mining the Organisational Perspective of Business Processes. Decis. Support Syst. **89**, 87–97 (2016)
[25] Shen, Q., Polyvyanyy, A., Lipovetzky, N., Kampik, T.: Agent System Event Data: Concepts, Dimensions, Applications. In: ER, pp. 56–72 (2024)
[26] Shen, Q., Polyvyanyy, A., Lipovetzky, N., Kampik, T.: Applying Organizational Mining to Discover Agent Systems from Event Data. Inf. Syst. **138**, 102669 (2026)
[27] Song, M., van der Aalst, W.M.P.: Towards Comprehensive Support for Organizational Mining. Decis. Support Syst. **46**, 300–317 (2008)
[28] Tour, A., Polyvyanyy, A., Kalenkova, A.A.: Agent System Mining: Vision, Benefits, and Challenges. IEEE Access **9**, 99480–99494 (2021)
[29] Tour, A., Polyvyanyy, A., Kalenkova, A.A., Senderovich, A.: Agent Miner: An Algorithm for Discovering Agent Systems from Event Data. In: BPM, pp. 284–302 (2023)
[30] Wooldridge, M.J., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-oriented Analysis and Design. Auton. Agents Multi Agent Syst. **3**, 285–312 (2000)
[31] Yang, J., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M., Yu, Y.: *OrdinoR*: A Framework for Discovering, Evaluating, and Analyzing Organizational Models Using Event Logs. Decis. Support Syst. **158**, 113771 (2022)