Using Interface Patterns for Compositional Discovery of Distributed System Models[1]

R.A. Nesterov <ranesterov@edu.hse.ru>
I.A. Lomazova <ilomazova@hse.ru>
National Research University Higher School of Economics,
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. Process mining offers various tools for studying process-aware information systems. They mainly involve several participants (or agents) managing and executing operations on the basis of process models. To reveal the actual behavior of agents, we can use process discovery. However, for large-scale processes, it does not yield models, which help understand how agents interact since they are independent and their concurrent implementation can lead to a very sophisticated behavior. To overcome this problem, we propose interface patterns, which allow getting models of multi-agent processes with a clearly identified agent behavior and interaction scheme as well. The correctness of patterns is provided via morphisms. We also conduct a preliminary experiment, results of which are highly competitive compared to the process discovery without interface patterns.

Keywords: Petri nets; interface patterns; synchronization; composition; morphisms; process discovery; multi-agent systems; distributed systems.

DOI: 10.15514/ISPRAS-2017-29(4)-2

For citation: Nesterov R.A., Lomazova I.A. Using Interface Patterns for Compositional Discovery of Distributed System Models. *Trudy ISP RAN/Proc. ISP RAS*, vol. 29, issue 4, 2017, pp. 21-38. DOI: 10.15514/ISPRAS-2017-29(4)-2

1. Introduction

Process mining is the relatively new direction in studying process-aware information systems. They include information systems managing and executing operational processes, which involve people, applications and information resources through process models [1]. Examples of these systems include workflow management systems, business process management systems, and enterprise information systems.

[1]This work is supported by the Basic Research Program at the National Research University Higher School of Economics and Russian Foundation for Basic Research, project No.16-01-00546.

The underlying interactions among participants (also called agents) of process-aware information systems are intrinsically distributed multiagent systems. An agent acts autonomously, but it can interact with the others via shared resources, restrictions, and other means. Process mining helps to extract a model of this system for further study from a record of its implementation called an event log. However, extracted models are hard for analysis since there might be complex interactions among process participants the number of those can be significant.

In this paper, we propose a compositional approach to address this problem. Given an event log of a distributed system, we can filter it by agents and mine a model of each agent. Then, agent models can be composed to get a complete model of a multi-agent distributed system, which might be simulated. Composing agent models allows us to obtain more structured models compared to models extracted from complete logs since the behavior of an agent can be clearly identified. We compose agent models via interface patterns, which describe how they intercommunicate. This approach was presented at TMPA-2017 [2], the conference proceedings will be available later. The formal proof of the composition correctness is based on using net morphisms [3]. Moreover, interface patterns allow us to inherit deadlock-freeness and proper termination from agents by construction.

We conduct a preliminary experiment on using one interface pattern for mining multi-agent models. The outcomes are evaluated with the help of conformance checking quality dimensions [1, 4] and complexity metrics proposed in [5].

This paper is structured as follows. The next section provides an overview of process discovery and compositional approaches. In Section 3 we introduce basic terms which are used in the paper. Section 4 shows a general description of the compositional approach to process discovery. Section 5 briefly introduces how we compose agent models using interface patterns and net morphisms. In Section 6 we describe the preliminary experiment and analyze results.

2. Related Work

There exist three types of process mining, namely discovery, conformance, and enhancement. Process discovery produces a process model out of an event log – a record of implemented activities. Existing discovery approaches can yield a model in a variety of notations including Petri nets, heuristic nets, process trees, BPMN, and EPC. Petri nets are the most widespread process model representations discovered from event logs. Conformance checking is used to check whether a discovered model corresponds to an input event log and to identify probable deviations. The main idea of enhancement is to improve existing processes using knowledge of actual processes (usually denoted AS-IS) obtained from event logs.

Process discovery offers several methods to be used for constructing models from event logs. One of the first and the most straightforward discovery approach is **α-algorithm,** which identifies ordering relations among activities in logs, but it has severe usage limitations connected with cycles and the overall quality of obtained models [1]. It has several refined versions and improvements, for example [6], but

there are other more sophisticated and efficient discovery algorithms. S. Leemans et al. [7] has proposed **inductive miner** allowing to extract process models from logs containing infrequent or incomplete behavior as well as dealing with activity lifecycle when there are separate actions of start and finish for each activity. Apart from that, inductive miner always produces well-structured models in the form of Petri nets. **HeuristicsMiner** is another process discovery algorithm proposed by A. Weijters et al. [8]. It can process event logs with a lot of noise (excessive activities) and also deals with infrequent process behavior. HeuristicsMiner uses intermediate casual matrices and produces heuristics net, which can easily be converted into Petri nets and applied for other notations including EPC, BPMN, and UML. S. van Zelst et al. [9] proposed the approach to process discovery based on **integer linear programming** and **theory of regions**. Their algorithm can produce Petri nets with complex control flow patterns, and its recent improvements guarantee the structural correctness of discovered models. C. Gunther and W. van der Aalst have proposed adaptive **fuzzy mining** approach [10] to deal with unstructured processes extracted from event logs since they can produce different abstractions of processes distinguishing "important" behavior.

Since state-of-the-art process discovery algorithms can deal with complex process behavior, the other problem is to obtain models that are appropriate concerning their structure. A good process model is readable and well-structured, i.e. there is no redundant elements or unnecessary structural complications. There is a so-called continuum of processes ranging from highly structured processes (Lasagna models) to unstructured processes (Spaghetti models) [1]. The problem of obtaining well-structured models is extensively studied in the literature. Researchers offer different techniques to improve model structure [11], and to produce already well-structured process models [12, 13, 14]. In the case of multi-agent and distributed systems using well-structured models should also allow us to identify agent behavior clearly for the model understandability improvement.

We suggest discovering models of agents independently and then composing them together to produce a structured multiagent system model with the clearly visible behavior of each agent. Several compositional approaches for process discovery have been proposed. In [15] A. Kalenkova et al. have shown how to obtain a more readable model from an event log by decomposing extracted transition systems. A special technique to deal with cancellations in process implementation and to produce clear and structured process models which can contain cancellations have been studied in [16]. Also, in [17] authors have proposed a technique for compositional process discovery based on localizing events using region theory to improve overall quality of discovered models.

Correct coordination of system components is an error-prone task. Their interaction can generate complex behavior. The majority of process discovery tools produce Petri nets, and a large amount of literature has investigated the problem of composing Petri nets. They can be composed via straightforward merging of places and transitions [18], but the composition result will not preserve component properties. One of the

possible ways to achieve inheritance of component behavioral properties is to use morphisms [19]. Special constructs for composing Petri net based on morphisms were studied in [3, 20, 21]. The key idea of this approach is that distributed system components refine an abstract interface describing the interactions between them. In [22] I. Lomazova has proposed a compositional approach for a flexible re-engineering of business process by using a system of interacting workflow nets. There also exists a several techniques for compositional synthesis of web services [23].

However, in [24] R. Hamadi and B. Benatallah have proposed an algebraic approach to the regular composition of services. These compositional approaches do not let specify the explicit order of inner behavior of two interacting components. This situation is schematically represented in Fig. 1. Having two discovered component models with always executable actions A and B, we want to require that they interact in a way that A is implemented before B. This way of intercommunication is also shown in the form of Petri net.

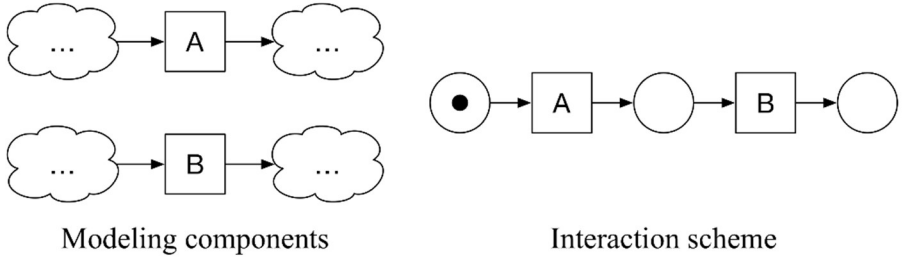

Fig.1. Defining relations on inner actions of components

In [2] we have proposed a solution to this problem and two other patterns for composing two interacting components. The obtained composition inherits properties, such as deadlock-freeness and proper termination, from components.

In this paper, we show how these patterns can be used for discovering a multi-agent system model from an event log in a compositional way. Applying compositional patterns allows us to obtain a more readable model improving time complexity due to the parallelization of process discovery.

We can assess process models obtained from event logs against four standard quality dimensions – fitness, precision, generalization, and simplicity [4]. Fitness identifies how accurately an extracted model can replay a source event log. Precision indicates a fraction of a behavior allowed by the model but not seen in the event log. Generalization tries to measure the extent to which the model will be able to implement the behavior of the process unseen so far in the log. Simplicity focuses on assessing structural complexity alongside with other graph characteristics – a number of elements and a structuredness measure [5].

3. Preliminary Definitions

3.1. Petri Nets

We use Petri nets [18] to represent agent models and an interaction scheme called interface.

Definition 1: A multiset m over a set S is a function $m: S \to \mathbb{N} \cup \{0\}$. Let m and m_0 be two multisets, $m_0 \subseteq m$ iff $\forall s \in S$: $m_0(s) \le m(s)$. Also, $\forall s \in S$: $(m+m_0)(s)=m(s)+m_0(s)$ and $(m-m_0)(s)=max(0, m(s), m_0(s))$.

Then, an ordinary set is a multiset in which distinct elements occur only once.

Definition 2: A Petri net is a bipartite graph $N=(P, T, F, m_0, L)$, where:

1. $P=\{p_1, p_2, ..., p_n\}$ – a finite non-empty set of places.
2. $T=\{t_1, t_2, ..., t_m\}$ – a finite non-empty set of transitions, $P \cap T=\emptyset$.
3. $F \subseteq (P \times T) \cup (T \times P)$ – a flow relation.
4. m_0: $P \to \mathbb{N} \cup \{0\}$ – a multiset over P, initial marking.
5. L: $T \to \mathcal{A} \cup \{\tau\}$ – a labeling for transitions, where τ is a name for silent transitions.

Pictorially, places are shown as circles, and transitions are shown as boxes (silent transitions are depicted by black boxes). A flow relation is depicted by directed arcs (see Fig. 2).

Let $X=P \cup T$. We call a set ${}^\bullet x=\{y \in X \mid (y,x) \in F\}$ a **preset** of x and a set $x^\bullet=\{y \in X \mid (x,y) \in F\}$ – a **postset** of x. *Also* ${}^\bullet x^\bullet={}^\bullet x \cup x^\bullet$ is a neighborhood of x.

The behavior of Petri nets is defined by the firing rule, which specifies when an action can occur, and how it modifies the overall state of the system.

A marking m: $P \to \mathbb{N} \cup \{0\}$ enables a transition t, denoted $m[t\rangle$, if ${}^\bullet t \subseteq m$. The t firing at m leads to m', denoted $m[t\rangle m'$, where $m'=m-{}^\bullet t+t^\bullet$. When $\forall t \in T$ *and* $\forall w \in T^*$, $m[tw\rangle m'=m[t\rangle m''[w\rangle m$, w is then called a firing sequence. We denote a set of all firing sequences of a net N as $FS(N)$.

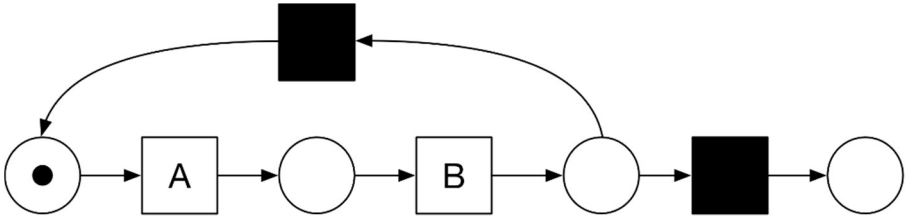

Fig.2. A Petri net with silent transitions

We call a marking m reachable from m_0 if $\exists w \in FS(N)$: $m_0[t\rangle m$. A set of all markings reachable from m_0 is denoted by $[m_0\rangle$. So, $[m\rangle$ is a set of all markings reachable from m. A net N is safe if $\forall p \in P$, $\forall m_0 \in [m_0\rangle$: $m(p) \le 1$.

A marking m_f is called final if $\forall p \in m_f$: $p^\bullet=\emptyset$. A net N is deadlock-free if $\forall t \in T$ $\exists m \in [m_0\rangle$: $m[t\rangle$ and $m \neq m_f$. A net N terminates properly if a final marking is reachable from all reachable states $\forall m \in [m_0\rangle$: $m_f \in [m\rangle$.

3.1. Event Logs

Process discovery techniques allow generating process models from event logs containing information on executed actions. In a simple case, event logs may contain actions names and a corresponding implementation order. We can augment this record with a timestamp (when an action occurs) and executor (what agent implements it).

Definition 3: Let $\mathcal{N}$ be a set of action names and $\mathcal{E}$ be a set of agent names. An activity is a triple (n, e, t), where $n \in \mathcal{N}$, $e \in \mathcal{E}$, and t corresponds to a timestamp. The set of all activities is denoted by Act. A trace $\sigma \in Act^+$ is a sequence of activities. An event log L is a multiset over Act^+, $L \in m(Act^+)$.

Different traces can be combined to form a case corresponding to a process implementation scenario. XES is a standard representation format adopted by IEEE [25] for logging events and processing them via process mining tools.

Table 1. A fragment of an event log

Trace ID	Action ID	Timestamp	Executor
Trace 1			
	t_1	2017-03-01T17:23:40	Agent 1
	e_2	2017-03-01T19:12:05	Agent 2
	…	…	…
Trace 2			
	e_1	2017-03-02T21:13:47	Agent 2
	t_1	2017-03-04T21:14:40	Agent 1
	…	…	…

4. Compositional Process Discovery

4.1. General Outline

To support the compositional discovery of models from event logs generated by multi-agent systems, we assume a record of each action has a corresponding label of an agent implementing it. The procedure of the compositional synthesis includes several steps to be implemented:

1. Capturing a complete event log L from multi-agent system operation.
2. Filtering the event log L by agent labels and producing a set of event logs L_e ($|L_e|=|\mathcal{E}|$), each trace consists of actions implemented by e only.

3. Discovering a model for each agent separately from the set of event logs L_e;
4. Defining interface pattern which describes how agents intercommunicate;
5. Composing agent models and producing a multi-agent system model.

The step of defining interface pattern for agent interaction is implemented manually so far. We rely on an expert view on how agents should intercommunicate.

4.2. Software Overview

A wide range of process discovery tools is implemented within the context of the open-source project **ProM** [26] continuously improving nowadays. However, there also exist many commercial tools using process mining approach to analyze and improve business process. They include **Disco** [27], **QPR ProcessAnalyzer** [28], **myInvenio** [29] to name but a few. Contrary to ProM, they provide more business-related solutions for process performance analysis and further improvement.

To process event logs we use the advanced ProM plugin **GENA** [30] which allows to generate event logs with timestamps and originator labels as well as to augment logs with artificial events representing noise.

5. Composing Petri Nets via Interface Patterns

This section provides a brief introduction to our approach to Petri net composition using interfaces and net morphisms.

5.1. Composing Petri Nets via morphisms

The notion of ω-morphism on Petri nets was first introduced in [3] for elementary net systems and can be applied for safe nets.

Definition 4: Let $N_i = (P_i, T_i, F_i, m_0^i, L_i)$ be two safe Petri nets for $i=1,2$. The ω-morphism is a total surjective map $\varphi: N_1 \to N_2$ such that:

1. $\varphi(P_1)=P_2$.
2. $\forall t_1 \in T_1: \varphi(t_1)\in T_2 \Rightarrow \varphi({}^\bullet t_1) = {}^\bullet\varphi(t_1) \wedge \varphi(t_1^\bullet) = \varphi(t_1)^\bullet$.
3. $\forall t_1 \in T_1: \varphi(t_1)\in P_2 \Rightarrow \varphi({}^\bullet t_1^\bullet) = \{\varphi(t_1)\}$.
4. $\forall p_1 \in P_1: m_0^1(p_1) > 0 \Rightarrow m_0^2(\varphi(p_1)) = m_0^1(p_1)$.

Figure 3 helps to explain requirements 2 and 3 of the definition. i.e. how transitions of N_1 can be mapped onto places and transitions of N_2.

To use morphisms for Petri net composition, we need to define morphisms from agent nets towards an interface net, which describes how they intercommunicate. Then we merge transitions having common labels and images. Figure 4 shows how two Petri nets are composed via ω-morphisms represented as dotted arrows.

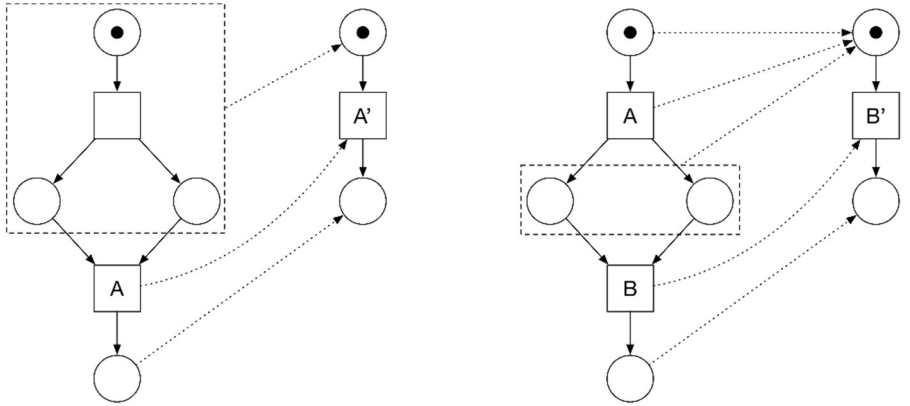

Fig.3. Transition map options for ω-morphism

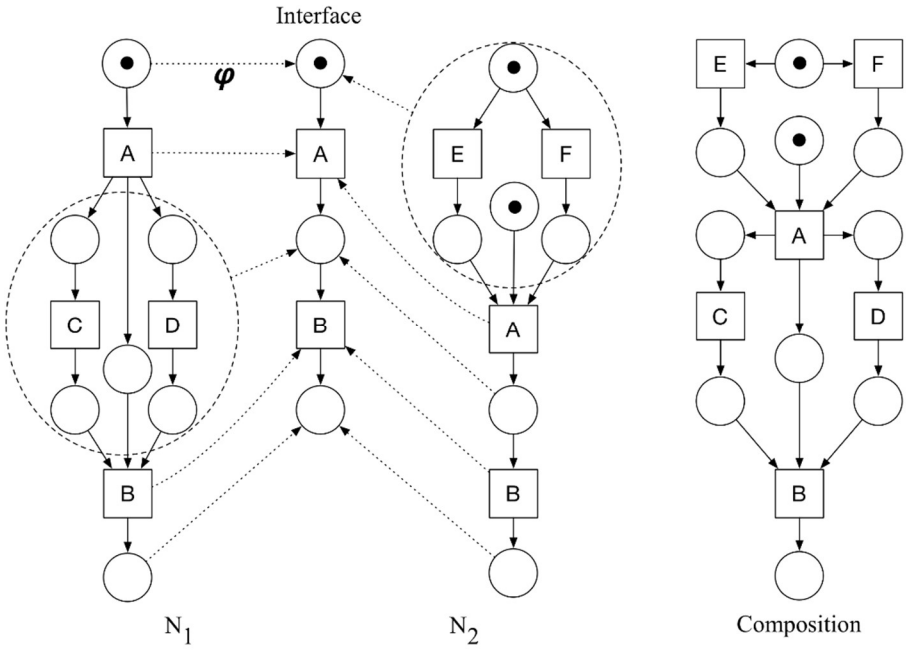

Fig.4. Composing two Petri nets via ω-morphism

As it was proved in [19], the use of morphisms allows us to preserve properties of interacting components in a composed process net. A composition obtained via ω-

morphisms is deadlock-free and properly terminates iff source component nets and interface net are deadlock-free and terminate properly as well.

5.2. Compositional Interface Patterns

To facilitate Petri net composition, we use compositional patterns for typical interface we have proposed in [2]. One of such patterns called the simple causality is schematically shown in Fig. 1, and Fig. 5 provides its instantiation. A pattern includes component and interface net which might be merged according to the morphism composition rules if there is a need to produce a model for comprehensive simulation.

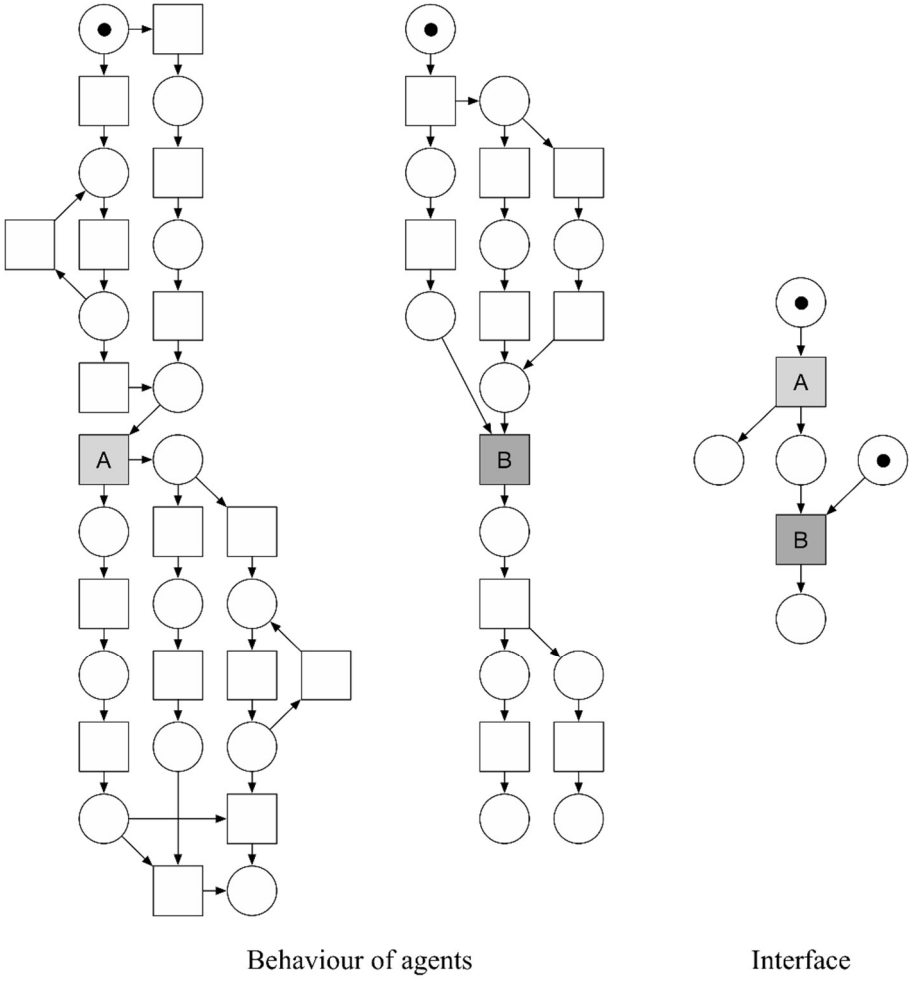

Fig.5. Instantiating simple causality pattern

It also has to be mentioned that to preserve concurrency in the implementation of interacting agents we expand interface nets with additional places and transitions keeping them weakly bisimilar with original interfaces. Consequently, extended interfaces allow us to obtain composition results with the clearly identified behavior of each component.

Figure 5(b) shows how we have expanded interface net for this pattern. We use expanded interfaces only for our inner purposes. The end user does not need to know the underlying theoretical aspects of our approach.

6. Some Experimental Evaluation

In this section, we describe a preliminary experiment on using the simple causality pattern for compositional process discovery. To test our approach we use artificial event logs obtained from the instantiated simple causality pattern. Then we also assess quality metrics of discovered models and provide a balanced consideration.

6.1. Processing Event Logs

Using GENA and the composition result obtained from the instantiated simple causality pattern (see Fig. 5) we have generated the event log with 3000 traces. Then we have filtered the initial log by executors using ProM. The obtained event logs have the characteristics presented in Table 2. Generation results for Agent A show bigger values due to cycles.

Table 2. Characteristics of event logs

	Log *L*	**Log L_A**	**Log L_B**
Number of traces	3000	3000	3000
Number of events	58466	34466	24000
Events per trace (min)	17	9	8
Events per trace (max)	43	35	8
Events per trace (mean)	19	11	8

6.2. Discovering a System Model from Log *L*

Figure 6 shows the fragment of the Petri net discovered from the event log L using Inductive Miner and ProM. The behavior of agents is distinguished by colors.

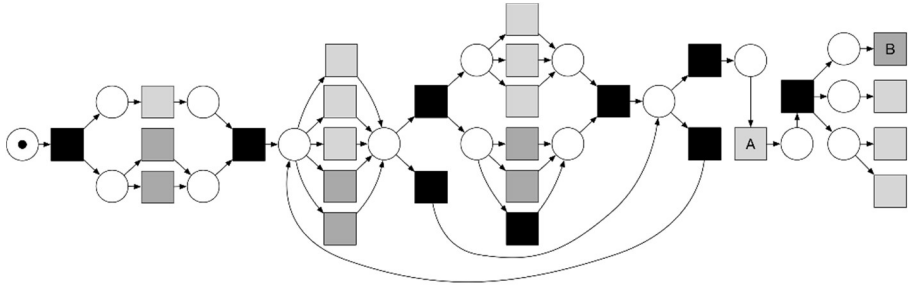

Fig.6. The fragment of the system model discovered from L

This discovered model is quite well-structured (constructed out of clear blocks) but it does not allow to identify the behavior of different agents. That is why, it is hard to yield the complete picture of agent intercommunication scheme.

6.3. Discovering and Composing Models from Logs L_A and L_B

Figure 7 shows the fragment of the composed Petri nets we have discovered from the agent event logs L_A and L_B also using Inductive Miner and ProM. It has to be mentioned that Petri nets discovered by Inductive Miner are always safe. Hence we can apply the approach based on morphisms to compose separately discovered models of agent behavior.

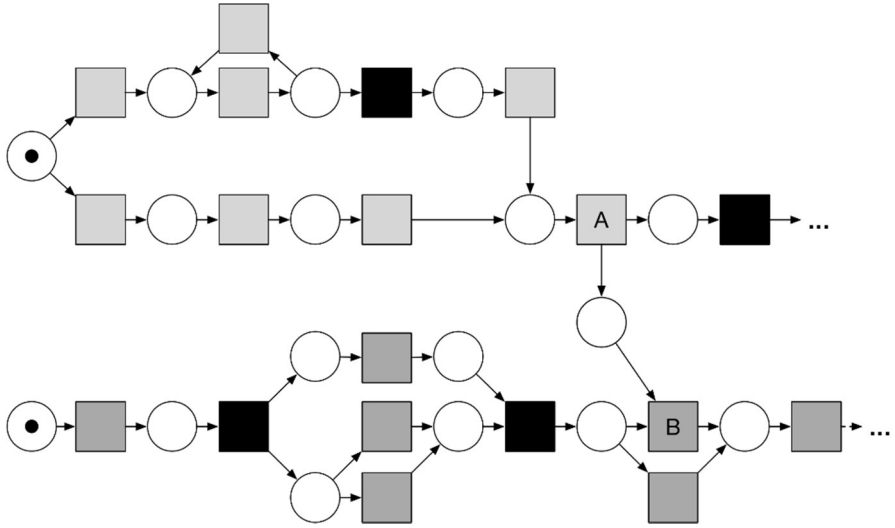

Fig.7. The fragment of the composed system model discovered from L_A and L_B

The merged model allows us to identify the behavior of agents clearly and how they intercommunicate. Using morphisms guarantees inheritance of properties such as deadlock-freeness and proper termination of agents by the entire net.

6.4. Analysis of the Experiment Results

ProM implementation of Inductive miner offers three configuration options:

1. event logs with **infrequent** behavior;
2. event logs with **incomplete** behavior;
3. event logs with **lifecycle** events (start/finish of events);
4. exhaustive k-successor algorithm.

We do not work with incomplete logs or with lifecycle logs for now. So, in our experiment we have discovered models of system and agents shown in previous subsections in accordance with options 1 and 4 and compared them using structural process discovery metrics.

Table 3 provides the comparison of structural characteristics for the directly discovered and composed system models. We have compared obtained models with respect to the number of Petri net elements and structure metric which assess the overall complexity of a model by breaking it into trivial constructs and assigning weights to each reducing step. Models discovered with infrequent configuration are denoted as INFR, models discovered with exhaustive configuration are denoted as EXHS.

The experiment results show the increase in transition numbers because of adding silent transitions. Compositional patterns obviously decrease a number of arcs, compared to direct discovery, as long as we simplify agent intercommunication. Composed models also preserve complex control flows as shown by structuredness measure. Separately discovered agent models and their composition exhibit more precise cycle discovery.

Table 3. Structural analysis of system models

		Direct		Composed	
	Source	**INFR**	**EXHS**	**INFR**	**EXHS**
Places	28	30	47	35	39
Transitions	27	44	46	40	41
Arcs	68	100	114	89	93
Structuredness	9360	240	496	872	1208

We have also conducted conformance checking for directly discovered and composed models. As it was mentioned above, there are four standard quality dimensions, namely fitness, precision, simplicity, and generalization. Simplicity is analyzed above via structural analysis. We do not estimate generalization since there are no complex cyclic or concurrent constructs to instantiate the simple causality pattern. Table 4 shows values obtained for fitness and precision of discovered and composed system models.

Table 4. Quality analysis of system models

		Direct		Composed	
	Source	**INFR**	**EXHS**	**INFR**	**EXHS**
Fitness	1,0000	1,0000	0,9684	1,0000	1,0000
Precision	0,6992	0,3631	0,5508	0,5629	0,6232

Both discovered and composed system models preserve the appropriate level of fitness, the composition does not block its preservation. What is more important, using compositional patterns produces models with precision nearer to that of the source model compared to direct discovery results. Composed models approximately 30% more precise than discovered ones.

To sum up, we used the simple causality pattern to produce the model of the multi-agent system. Assessment results showed that the composed models are highly competitive with the models directly discovered from complete event logs in the context of their relative structural complexity evaluations and conformance checking results.

7. Conclusion and Future Work

In this paper, we have proposed the solution to the problem of discovering structured models for the processes with several participants (agents). The key idea is to automatically obtain the correct and complete process models from the separate source models of its components. The interaction between agents is defined by experts.

To prove the correctness of the composition we adopt the approach based on Petri net morphisms. We refer to the compositional patterns proposed for the correct synthesis of models for multi-agent processes. In the context of this work, we conducted the preliminary experiment on using the simple causality pattern for constructing the complete model from discovered agent models. The analysis of experimental results (conformance and complexity) showed that composed models are highly competitive compared to the models obtained directly. Moreover, our compositional approach to process discovery allows producing models with the clearly identified behavior of interacting agents.

We aim to continue developing of compositional patterns for typical interfaces and providing experimental process discovery implementations for them using also real-live event logs. Also, we will proceed with complex synchronization patterns with relations on action sets and their correct combinations.

References

[1]. van der Aalst W.M.P. Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st ed. Springer Publishing Company, Incorporated, 2011. DOI: 10.1007/978-3-642-19345-3.

[2]. Nesterov R.A., Lomazova I.A. Compositional process model synthesis based on interface patterns. Communications in Computer and Information Science, 2017.

[3]. Bernardinello L., Mangioni E., Pomello L. Local state refinement and composition of elementary net systems: An approach based on morphisms. Transactions on Petri Nets and Other Models of Concurrency, 2013, vol. 8, pp. 48–70. DOI: 10.1007/978-3-642-40465-8_3.

[4]. Buijs J.C.A.M., Dongen B., van der Aalst W.M.P. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. Lecture Notes in Computer Science, 2012, vol. 7565, pp. 305–322. DOI: 10.1007/978-3-642-33606-5_19.

[5]. Lassen K.B., van der Aalst W.M.P. Complexity metrics for workflow nets. Information and Software Technology, 2009, vol. 51, issue 3, pp. 610–626. DOI: 10.1016/j.infsof.2008.08.005.

[6]. Wen L., van der Aalst W.M.P., Wang J., Sun J. Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery, 2007, vol. 15, issue 2, pp. 145–180. DOI: 10.1007/s10618-007-0065-y.

[7]. Leemans S.J.J., Fahland D., van der Aalst W.M.P. Discovering block-structured process models from event logs containing infrequent behavior. Lecture Notes in Business Information Processing, 2013, vol. 171, pp. 66–78. DOI: 10.1007/978-3-319-06257-0_6.

[8]. Weijters A.J.M.M., van der Aalst W.M.P., de Medeiros A.K.A. Process Mining with the HeuristicsMiner Algorithm. BETA Working Paper Series, 2006, vol. 166, Einhoven University of Technology.

[9]. van Zelst S.J., van Dongen B.F., van der Aalst W.M.P. ILP-based process discovery using hybrid regions. CEUR Workshop Proceedings, 2015, vol. 1731, pp. 47–61.

[10]. Gunther C.W., van der Aalst W.M.P. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. Lecture Notes in Computer Science, 2007, vol 4714, pp. 328–343. DOI: 10.1007/978-3-540-75183-0_24.

[11]. van der Aalst W.M.P., Gunther C.W. Finding structure in unstructured processes: The case for process mining. ACSD '07 Proceedings of the Seventh International Conference on Application of Concurrency to System Design, 2007, pp. 3–12. DOI: 10.1109/ACSD.2007.50

[12]. Buijs J.C.A.M. Flexible Evolutionary Algorithms for Mining Structured Process Models. Ph.D. dissertation, Eindhoven University of Technology, 2014.

[13]. Smedt J.D., Weerdt J.D., Vanthienen J. Multi-paradigm process mining: Retrieving better models by combining rules and sequence. Lecture Notes in Computer Science, 2014, vol. 8841, pp. 446–453. DOI: 10.1007/978-3-662-45563-0_26.

[14]. de San Pedro J., Cortadella J. Mining Structured Petri Nets For The Visualization Of Process Behavior. SAC '16 Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 839–846. DOI: 10.1145/2851613.2851645.

[15]. Kalenkova A.A., Lomazova I.A., van der Aalst W.M.P. Process Model Discovery: A Method Based on Transition System Decomposition. Lecture Notes in Computer Science, 2014, vol. 8489, pp. 71–90. DOI: 10.1007/978-3-319-07734-5_5

[16]. Kalenkova A.A., Lomazova I.A. Discovery of Cancellation Regions within Process Mining Techniques. Fundamenta Informaticae, 2014, vol. 133, issue 2-3, pp. 197–209. DOI: 10.3233/FI-2014-1071.

[17]. van der Aalst, Kalenkova A., Rubin V., Verbeek E. Process Discovery Using Localized Events. Lecture Notes in Computer Science, 2015, vol. 9115, pp. 287–307. DOI: 10.1007/978-3-319-19488-2_15.

[18]. Reisig W. Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013, 145 p. DOI: 10.1007/978-3-642-33278-4.
[19]. Winskel G. Petri nets, morphisms and compositionality. Lecture Notes in Computer Science, 1985, vol. 222, pp. 453–477. DOI: 10.1007/BFb0016226.
[20]. Bernardinello L., Monticelli E., Pomello L. On Preserving Structural and Behavioural Properties by Composing Net Systems on Interfaces. Fundamenta Informaticae, 2007, vol. 80, issue 1-3, pp. 31–47.
[21]. Bernardinello L., Pomello L., Scaccabarozzi S. Morphisms on Marked Graphs. CEUR Workshop Proceedings, 2014, vol. 1160, pp. 113–127.
[22]. Lomazova I.A. Interacting Workflow Nets For Workflow Process Reengineering. Fundamenta Informaticae, 2010, vol. 101, issue 1-2, pp. 59–70. DOI: 10.3233/FI-2010-275.
[23]. Cardinale Y., Haddad J.E., Manouvrier M., Rukoz M. Web Service Composition Based On Petri Nets: Review and Contribution. Lecture Notes in Computer Science, vol. 8194, 2012, pp. 83–122. DOI: 10.1007/978-3-642-45263-5_5.
[24]. Hamadi R., Benatallah B. A Petri Net-Based Model For Web Service Composition. ADC'03 Proceedings of the 14th Australasian database conference, 2003, pp. 191–200.
[25]. XES (eXtensible Event Stream). Available at: http://www.processmining.org/logs/xes, accessed 10.03.2017.
[26]. van Dongen B.F., de Medeiros A.K.A., Verbeek H.M.W., Weijters A.J.M.M., van der Aalst W.M.P. The ProM Framework: A New Era in Process Mining Tool Support. Lecture Notes in Computer Science, 2005, vol. 3536, pp. 444–454. DOI: 10.1007/11494744_25.
[27]. Gunther C.W., Rozinat A. Disco: Discover your processes. CEUR Workshop Proceedings, 2012, vol. 940, pp. 40–44.
[28]. Ailenei I.M. Process Mining Tools: A Comparative Analysis. Master's thesis, Eindhoven University of Technology, 2011.
[29]. BPM Tool – myInvenio. Available: https://www.my-invenio.com, accessed 10.03.2017.
[30]. Shugurov I.S., Mitsyuk A.A. Generation of a Set of Event Logs with Noise. Proceedings of the 8th Spring/Summer Young Researchers Colloquium on Software Engineering (SYRCoSE 2014), 2014, pp. 88–95.DOI: 10.15514/SYRCOSE-2014-8-13.

Автоматизированный композициональный синтез моделей распределенных систем с помощью паттернов интерфейсов

Р.А. Нестеров <ranesterov@edu.hse.ru>
И.А. Ломазова <ilomazova@hse.ru>
Научно-учебная лаборатория процессно-ориентированных информационных систем (ПОИС),
Национальный исследовательский университет «Высшая школа экономики»
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Средства и методы process mining позволяют исследовать различные аспекты процессно-ориентированных информационных систем. Как правило, в рамках таких систем несколько исполнителей (агентов) взаимодействуют друг с другом. Поведение агентов, а также механизмы их взаимодействия описываются с помощью моделей процессов. Для моделирования процессов мы применяем обыкновенные сети Петри. Алгоритмы process discovery позволяют восстановить модели реального поведения агентов из журнала событий системы. Однако в случае масштабных систем анализ взаимодействия как поведения отдельных агентов, так и всей системы в целом затруднителен, так как получаемые модели крупномасштабных систем в большинстве случаев крайне громоздкие и плохо читаемые. Для решения этой проблемы мы предлагаем использовать так называемые паттерны интерфейсов, которые описывают, как агенты взаимодействуют друг с другом. С их помощью полная модель мультиагентной системы может быть получена путем композиции отдельных моделей агентов. Кроме того, модели мультиагентных систем, построенные с применением паттернов интерфейсов, позволяет легко идентифицировать поведение каждого отдельного агента. В целях обеспечения корректности применения паттернов интерфейсов мы применяем специальные конструкции на сетях Петри – морфизмы. Результаты эксперимента по применению паттерна для композиционального синтеза модели мультиагентной системы, представленные в работе, показали прирост основных метрик качества по сравнению с моделями, получаемыми с помощью стандартного подхода process discovery.

Ключевые слова: сети Петри; паттерны интерфейсов; синхронизация; композиция; морфизмы; извлечение процессов; мультиагентные системы; распределенные системы.

DOI: 10.15514/ISPRAS-2017-29(4)-2

Для цитирования: Нестеров Р.А., Ломазова И.А. Автоматизированный композициональный синтез моделей распределенных систем с помощью паттернов интерфейсов. *Труды ИСП РАН*, том 29, вып. 4, 2017 г., стр. 21-38 (на английском языке). DOI: 10.15514/ISPRAS-2017-29(4)-2

Список литературы

[1]. van der Aalst W.M.P. Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st ed. Springer Publishing Company, Incorporated, 2011. DOI: 10.1007/978-3-642-19345-3.

[2]. Nesterov R.A., Lomazova I.A. Compositional process model synthesis based on interface patterns. Communications in Computer and Information Science, 2017.

[3]. Bernardinello L., Mangioni E., Pomello L. Local state refinement and composition of elementary net systems: An approach based on morphisms. Transactions on Petri Nets and Other Models of Concurrency, 2013, vol. 8, pp. 48–70. DOI: 10.1007/978-3-642-40465-8_3.

[4]. Buijs J.C.A.M., Dongen B., van der Aalst W.M.P. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. Lecture Notes in Computer Science, 2012, vol. 7565, pp. 305–322. DOI: 10.1007/978-3-642-33606-5_19.

[5]. Lassen K.B., van der Aalst W.M.P. Complexity metrics for workflow nets. Information and Software Technology, 2009, vol. 51, issue 3, pp. 610–626. DOI: 10.1016/j.infsof.2008.08.005.

[6]. Wen L., van der Aalst W.M.P., Wang J., Sun J. Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery, 2007, vol. 15, issue 2, pp. 145–180. DOI: 10.1007/s10618-007-0065-y.

[7]. Leemans S.J.J., Fahland D., van der Aalst W.M.P. Discovering block-structured process models from event logs containing infrequent behavior. Lecture Notes in Business Information Processing, 2013, vol. 171, pp. 66–78. DOI: 10.1007/978-3-319-06257-0_6.

[8]. Weijters A.J.M.M., van der Aalst W.M.P., de Medeiros A.K.A. Process Mining with the HeuristicsMiner Algorithm. BETA Working Paper Series, 2006, vol. 166, Einhoven University of Technology.

[9]. van Zelst S.J., van Dongen B.F., van der Aalst W.M.P. ILP-based process discovery using hybrid regions. CEUR Workshop Proceedings, 2015, vol. 1731, pp. 47–61.

[10]. Gunther C.W., van der Aalst W.M.P. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. Lecture Notes in Computer Science, 2007, vol 4714, pp. 328–343. DOI: 10.1007/978-3-540-75183-0_24.

[11]. van der Aalst W.M.P., Gunther C.W. Finding structure in unstructured processes: The case for process mining. ACSD '07 Proceedings of the Seventh International Conference on Application of Concurrency to System Design, 2007, pp. 3–12. DOI: 10.1109/ACSD.2007.50

[12]. Buijs J.C.A.M. Flexible Evolutionary Algorithms for Mining Structured Process Models. Ph.D. dissertation, Eindhoven University of Technology, 2014.

[13]. Smedt J.D., Weerdt J.D., Vanthienen J. Multi-paradigm process mining: Retrieving better models by combining rules and sequence. Lecture Notes in Computer Science, 2014, vol. 8841, pp. 446–453. DOI: 10.1007/978-3-662-45563-0_26.

[14]. de San Pedro J., Cortadella J. Mining Structured Petri Nets for the Visualization of Process Behavior. SAC '16 Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 839–846. DOI: 10.1145/2851613.2851645.

[15]. Kalenkova A.A., Lomazova I.A., van der Aalst W.M.P. Process Model Discovery: A Method Based on Transition System Decomposition. Lecture Notes in Computer Science, 2014, vol. 8489, pp. 71–90. DOI: 10.1007/978-3-319-07734-5_5.

[16]. Kalenkova A.A., Lomazova I.A. Discovery of Cancellation Regions within Process Mining Techniques. Fundamenta Informaticae, 2014, vol. 133, issue 2-3, pp. 197–209. DOI: 10.3233/FI-2014-1071.

[17]. van der Aalst, Kalenkova A., Rubin V., Verbeek E. Process Discovery Using Localized Events. Lecture Notes in Computer Science, 2015, vol. 9115, pp. 287–307. DOI: 10.1007/978-3-319-19488-2_15.

[18]. Reisig W. Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013, 145 p. DOI: 10.1007/978-3-642-33278-4.

[19]. Winskel G. Petri nets, morphisms and compositionality. Lecture Notes in Computer Science, 1985, vol. 222, pp. 453–477. DOI: 10.1007/BFb0016226.

[20]. Bernardinello L., Monticelli E., Pomello L. On Preserving Structural and Behavioural Properties by Composing Net Systems on Interfaces. Fundamenta Informaticae, 2007, vol. 80, issue 1-3, pp. 31–47.

[21]. Bernardinello L., Pomello L., Scaccabarozzi S. Morphisms on Marked Graphs. CEUR Workshop Proceedings, 2014, vol. 1160, pp. 113–127.

[22]. Lomazova I.A. Interacting Workflow Nets For Workflow Process Reengineering. Fundamenta Informaticae, 2010, vol. 101, issue 1-2, pp. 59–70. DOI: 10.3233/FI-2010-275.

[23]. Cardinale Y., Haddad J.E., Manouvrier M., Rukoz M. Web Service Composition Based On Petri Nets: Review and Contribution. Lecture Notes in Computer Science, vol. 8194, 2012, pp. 83–122. DOI: 10.1007/978-3-642-45263-5_5.

[24]. Hamadi R., Benatallah B. A Petri Net-Based Model For Web Service Composition. ADC‘03 Proceedings of the 14th Australasian database conference, 2003, pp. 191–200.

[25]. XES (eXtensible Event Stream). Available at: http://www.processmining.org/logs/xes, accessed 10.03.2017.

[26]. van Dongen B.F., de Medeiros A.K.A., Verbeek H.M.W., Weijters A.J.M.M., van der Aalst W.M.P. The ProM Framework: A New Era in Process Mining Tool Support. Lecture Notes in Computer Science, 2005, vol. 3536, pp. 444–454. DOI: 10.1007/11494744_25.

[27]. Gunther C.W., Rozinat A. Disco: Discover your processes. CEUR Workshop Proceedings, 2012, vol. 940, pp. 40–44.

[28]. Ailenei I.M. Process Mining Tools: A Comparative Analysis. Master’s thesis, Eindhoven University of Technology, 2011.

[29]. BPM Tool – myInvenio. Available: https://www.my-invenio.com, accessed 10.03.2017.

[30]. Shugurov I.S., Mitsyuk A.A. Generation of a Set of Event Logs with Noise. Proceedings of the 8th Spring/Summer Young Researchers Colloquium on Software Engineering (SYRCoSE 2014), 2014, pp. 88–95. DOI: 10.15514/SYRCOSE-2014-8-13.