Modelling and Validation of Trading and Multi-Agent Systems: An Approach Based on Process Mining and Petri Nets*

Julio C. Carrasquel and Irina A. Lomazova

National Research University Higher School of Economics,
Myasnitskaya ul. 20, 101000 Moscow, Russia
jcarrasquel@hse.ru, ilomazova@hse.ru

Abstract. This paper presents our research on trading and multi-agent systems. Trading systems support the processes of buying/selling financial instruments between traders, so the validation of their correctness is a crucial task. Conversely, multi-agent systems is a current topic of interest within the analysis of interactive processes. We use Petri nets as the formalism for system modelling and simulation, whereas for validation we consider the use of process mining, and specifically conformance checking. Our research aims to use and develop conformance heuristics that can be aware on the *data perspective* of processes, and to take into account *concurrent and non-isolated process instances* whose execution may depend on each other.

Keywords: trading systems, multi-agent systems, petri nets, process mining, conformance checking.

1 Introduction

This position paper presents our research on the modelling and validation of trading and multi-agent systems.Trading systems support the processes of trading financial instruments (also known as securities). Since most of these systems are automated, traders submit orders with data attributes specifying what to buy or sell, and on what terms; many classes of orders exist allowing the traders to configure orders according to their trading strategy [11]. We aim to diagnose whether some process instances, handling orders from traders, *deviate* from their expected behavior; thus, we would like to answer questions of the following nature: Do all orders of a given class strictly follow their associated rules? If so, which are the deviating orders? From which traders? Here, when checking compliance of a case, it is needed to be aware of the order data attributes in such a way to correctly assess whether the violation of some rule has occurred.

* This work is supported by the Basic Research Program at the National Research University Higher School of Economics.

Conversely, multi-agent systems refers to a kind of system where *agents* interact within some environment; to assess whether an agent (a process instance) may deviate from its expected behavior may depend on different factors: the inner state of the agent, the states of other agents, or the current condition of the environment (also referred to as a *context* or as a system *nesting* the agents); thus, when checking compliance of an agent, it is needed to be aware of other process instances and the environment with which such an agent interacts.

We propose to work with process mining [6] to check deviations in both scenarios. Unlike traditional data mining techniques, which are process agnostic, process mining aims to discover, diagnose and improve processes given some observed behavior (i.e., event logs) and some formal description of the expected behavior (i.e, process models, business rules). Within process mining, we consider conformance checking [8]; conformance checking techniques aim to detect whether some process instances, captured from event logs, deviate from their expected behavior (described by some process model); however, state-of-the-art conformance checking techniques mostly focus on the control-flow, i.e., causal dependence between activities. Instead, we need to come up with more clever heuristics in our research: In trading systems is it important to consider *data-aware* conformance checking (to consider data attributes), whereas in multi-agent systems it is needed to reason on *multiple-instance-aware* conformance checking (to consider the interaction between process instances). Checking different perspectives on conformance checking is a current issue in which the process mining community has been driving its efforts; thus, this research aims to provide a contribution in this area.

In the remainder of this paper, we present our research as follows. Section 2 presents the research approach. Sections 3 presents our initial work addressing the domains of trading and multi-agent systems. Section 4 presents some related literature. Finally, section 5 provides the conclusions.

2 The Research Approach

The use and development of conformance heuristics, data-aware and for multi-agent systems, are our ultimate goals. Thus, there are other activities to carry out thorughout our work. Fig. 1 presents the research overall scheme. We distinguish the following main activities throughout our work:

Formal Modelling: To model processes we use formalisms with a convenient visualization and clear semantics. We use Petri nets [20] — a formalism for modelling concurrent systems with a strong theoretical basis, and a wide range of analysis techniques; Existing high-level classes of Petri nets allow, for the use of data attributes, i.e., coloured Petri nets [12]; for the modelling of multi-agent systems, i.e., nested Petri nets [15]; and more recently, for connecting process models with non-local data structures, i.e, DB-nets [19]. A model can be also a set of logic formulae, i.e, based on *temporal logics*, stating a specific kind of behavior to be guaranteed in the system. The formalisms we use are aimed to be *normative* since these describe the behavior that the system must comply.

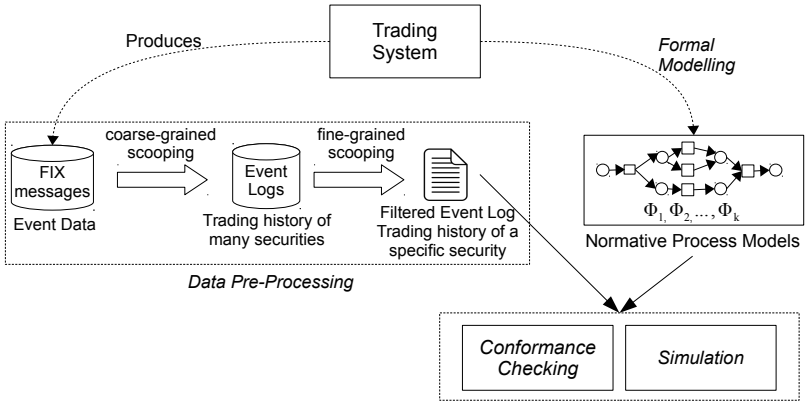

Fig. 1: The overall research approach.

Data Pre-Processing: Users communicate with a trading system via different interfaces, i.e, the Financial Information Exchange (FIX) protocol [2] is a widely used standard to enable such communication. As an input for process mining, we consider samples of FIX messages. These messages encode, for instance, the connection management, submission of orders to trade, system reports, etc; many processes can be extracted from such messages: the process of maintaining a connection, the process of trading an order, etc. We refer as *coarse-grained scooping* the extraction, from the FIX messages, an event log (a set of cases) of a process of interest. In particular, we are interested in the process of handling an order, i.e., from the moment a user submits an order, until it goes out from the system. The obtained event log may be refined (*fine-grained scooping*) to take just cases of interest, i.e, cases where orders are trading the same security. We plan to further discuss about this procedure of data pre-processing in upcoming works.

Conformance Checking and Simulation: Given some model describing a system process we aim to simulate it to get insights regarding its functioning. We consider consolidated Petri net tools such as CPN Tools [1] or Renew [13]. On such model, we also aim to replay an event log corresponding to the same process. Ultimately, our goal is to develop data- and multiple-instance-aware conformance checking heuristics that can provide diagnostics on possible deviations of cases observed in the event log, and that do not comply the norms described in the models.

3 Research Application Domains

3.1 Trading Systems

We focus on electronic trading systems, i.e, where trades are automatically performed using orders [11]; users submit orders in the system to either buy or sell securities; an order is an object $o = (id, sec, p, q, s)$, with an identifier id, specifying the security sec being traded, the price p per stock, the number of stocks

Table 1: Arrival of orders trading the same security, insertion in their order book (based on a price-time priority), and afterwards, the execution of trades.

(a) submitted orders

time	order	price	size	side
10:01	o_{B1}	20.0	4	*buy*
10:02	o_{S1}	22.0	2	*sell*
10:03	o_{S2}	22.0	4	*sell*
10:04	o_{B2}	22.2	5	*buy*

(b) order book initial state

buyer	size	price	size	seller
		22.0	4	o_{S2}
o_{B2}	5	22.0	2	o_{S1}
o_{B1}	4	20.0		

(c) executed trades

#	buyer	seller	quantity
1	o_{B2}	o_{S1}	2
2	o_{B2}	o_{S2}	3

q, and a value $s \in \{buy, sell\}$ indicating whether the user is buying or selling. Such structure is just for example purposes; orders can have many more complex attributes allowing the user to configure the trading according to some strategy. Upon their submission, orders may be placed in an *order book* — a two-sided priority queue where orders are served according to some priority. Each order book is related with a unique security; typically, the priority is a price-time policy: orders whose prices are the *best*[1] are served first, and if two orders have the same price, it is served first the order submitted earlier (see table 1). The first order on each side of the book is also referred to the *highest ranked order*. A match between two orders may occur as long as the price of the highest ranked buy order is greater or equal than the price of the highest ranked sell order. For instance, after the execution of the trades depicted in table 1(c). no trade execution will be possible since the price of the next best buy order o_{B1} is not greater or equal than the price for the remaining quantity of the next best sell order o_{S1}.

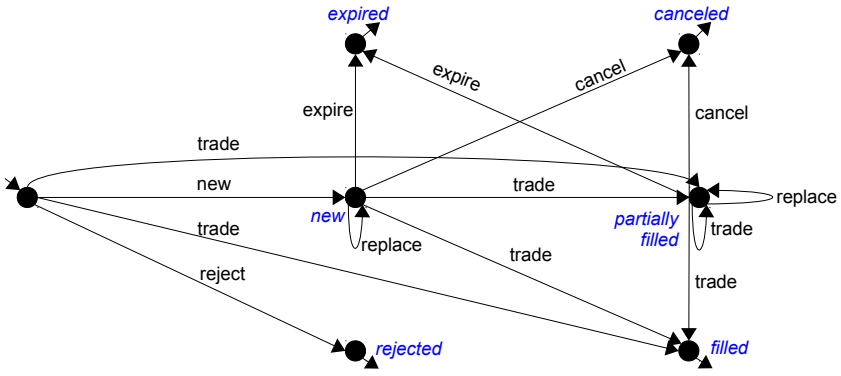

Fig. 2: Order handling process as a *labelled transition system*. The node with a small inbound arrow represents the initial state, whereas nodes with small outbound arrows represent the final states.

We consider the process related to the handling of an order, i.e, from an initial event where an order is submitted into the system up to a final event

[1] In the buy side, the best price is from the buy order whose price is the maximum, whereas as in the sell side, the best price is from the sell order whose price is the minimum.

where the order is discarded. Fig. 2 depicts the order handling process as a *labelled transition system* — the nodes represent the state of an individual order, whereas the transitions denote the activity fired over such order[2]. We extract cases related to this process from a sample of FIX messages, as described in section 2. Table 2 presents a small fragment from a real event log generated from the FIX messages where each case, and its corresponding set of events, is related with an order identifier (order id), thereby keeping track of what happens with each order and its attributes. All orders (case) of this event log trades the same security, so these belong to a same order book. As an example, fig. 3 provides a synthesized view of the observed behavior of each order in that event log.

Table 2: Fragment from a real event log related to handling process of distinct orders in a trading system.

order id	event id	activity	state	timestamp	price	size	side
T272	1	new	*new*	18-02-2019T05:50:49.382	10	500	*sell*
T272	2	trade	*filled*	18-02-2019T05:50:49.391	10	0	*sell*
T273	3	trade	*partially filled*	18-02-2019T05:50:49.391	10	500	*buy*
T273	4	replace	*partially filled*	18-02-2019T05:50:49.406	10	1000	*buy*
T273	5	cancel	*canceled*	18-02-2019T05:50:49.925	10	1000	*buy*

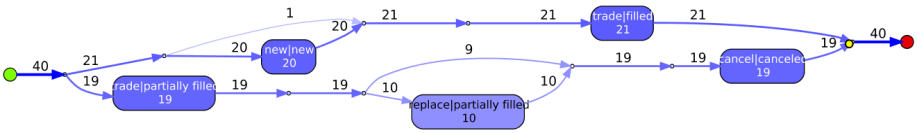

Fig. 3: Observed behavior from a real event log consisting of 40 orders (cases) trading a specific security — synthesized using ProM's inductive visual miner.

Orders are non-isolated, i.e, a buy order depends on a sell order to be filled and vice-versa; thus, we aim to visualize orders interacting in a single simulation model displaying the order book dynamics. To this aim, input event logs can be suitably manipulated and further processed, changing the perspective of what is the process, i.e, a case could be now understood as the trading session on a specific order book, whereas the events may have the orders as their attributes. Events that actually refer to the same action (i.e, in table 2, events 2 and 3 refer to the same trade execution) can be merged. Such further processed event log may be replayed on a high-level Petri net simulation model, for instance, using CPN Tools or the Renew platform. The semantics of the selected modelling language and the supporting tool should allow to model the order book structure, as well as to formulate its priority scheme.

[2] The *labelled transition system* in fig. 2 is an abstraction as there are more possible states and activities executed over an order. As an example, we refer the reader to the process flow of an order in the London Stock Exchange system [3].

After the reconstruction of the order handling processes from event logs and the simulation of the order book dynamics, we aim to focus to check whether some orders deviate from its expected behavior. For example, fig. 2 does not capture all the needed pre- and post-conditions while an order moves from a state to another; thus, order data attributes are also important to understand the correct behavior of each order, i.e, an order expires if some time has elapsed, the *trade* activity goes from the *partially filled* state to the *filled* state just if the order quantity of stocks goes to zero, etc. This leads us to research adequate data-aware conformance checking techniques for this scenario.

3.2 Multi-Agent System Processes

We define a type of agent as a class of process, i.e., the handling of a trading order, a customer in a bank, etc. We consider agents with well-defined initial and final states such that each agent: (i) it is instantiated upon request, (ii) it executes internal activities, either independently or influenced by an external control, (iii) it interacts with other agents, and (iv) it is terminated upon completion of its tasks. We model agents with workflow nets (WF-nets) [4] — a Petri net class to model business processes. However, WF-nets can hardly model at the same time a large number of process instances or their interaction.

We propose to model multi-agent systems (MAS) with nested Petri nets (NP-nets) [15] where tokens can be Petri nets themselves. A NP-net is constituted by a system net representing an environment, where agents may be created, interact, and eventually terminate; places in the system net store these agents as tokens with an inner Petri net structure (referred to as net tokens). NP-nets can model nested processes where the system net may be a parent process, i.e., a trading session, a bank customer service system, whereas net tokens may represent child processes, i.e., the handling of a single trading order, a customer, a bank employee; in the following, we present a formal definition of NP-nets, adapted from their classical definition, aimed to be used on the development of a *multiple-instance-aware* conformance heuristic.

Definition 1. *Let $Type$ be a set of agent types; Var — a set of typed (over $Type$) variables. We define a multi-agent system as a nested Petri net $NP = (SN, \{N_1, .., N_k\}, A, l)$ where:*

- $\forall_{i \in \{1,..,k\}}\ N_i = (P_i, T_i, F_i)$ *is a WF-net. Without loss of generality, we shall assume that $Type = \{N_1, ..., N_k\}$;*
- $SN = (P_{SN}, T_{SN}, F_{SN}, v, W)$ *— a system net where:*
  - P_{SN}, T_{SN}, F_{SN} *— the sets of places, transitions, and the flow relation;*
  - $v : P_{SN} \to Type$ *— a place-typing function;*
  - $W : F_{SN} \to Var$ *— an arc-labelling function, s.t for an arc r adjacent to a place p, the type of $W(r)$ coincides with the type of p.*
- *A is a finite set of activity labels. We consider the subset $\Lambda = \{\lambda_1, \lambda_2, ...\}$, $\Lambda \subseteq A$ as a set of labels used to synchronize the agents and the system net.*
- $l : (T_{SN} \cup T_1 \cup ... \cup T_k) \to A$ *is a transition labelling function.*

NP-nets provide four kind of steps to synchronize net tokens and the system net: (i) a *transport step* — the firing of a transition $t \in T_{SN}$ in the system net, $l(t) \notin \Lambda$, which *transports* net tokens across places, without changing their inner state; (ii) an *agent-autonomous step* — the firing of $t \in T_i$, $l(t) \notin \Lambda$, in a net token with structure N_i; (iii) a *horizontal synchronization step* — the firing of $t \in T_i$ and $t' \in T_j$ (in two distinct net tokens), lying in a same place of the SN, $l(t) = l(t') = \lambda$, $\lambda \in \Lambda$; and (iv) a *vertical synchronization step* — the firing of $t \in T_{SN}$, $l(t) = \lambda$, and multiple enabled transitions in some net tokens, provided that these transitions are *labelled* by $\lambda \in \Lambda$. Each of these net tokens are in a *binding* enabling the firing of t.

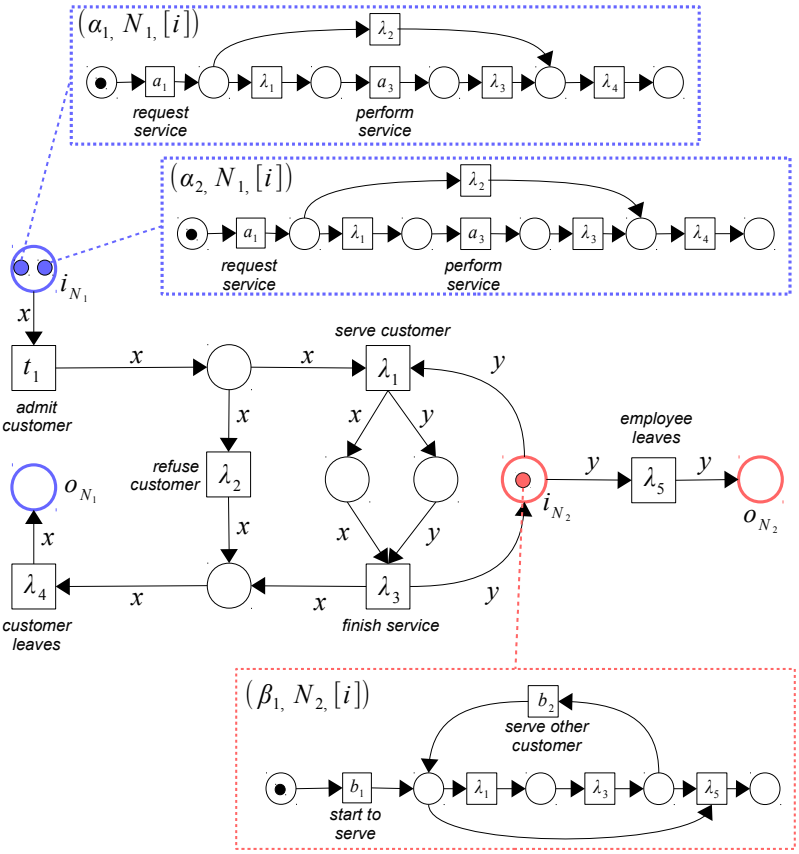

Fig. 4: A nested Petri net modelling a customer service system in a bank.

We assume a *stable* set of identified net tokens $\mathcal{N} = \{\alpha_1, \alpha_2, ...\}$ s.t. each agent $\alpha \in \mathcal{N}$ is a triple $\langle id, N_i, m \rangle$ where id is a net token identifier, $N_i \in \{N_1, ..., N_k\}$ is the inner structure of the net token, and m — a marking in N_i. This is referred to as a *strictly conservative* nested Petri net [16] where no net tokens are created, copied or disappeared, i.e, the cardinality of $\mathcal{N}$ is fixed.

Fig. 4 shows an example of a (*strictly conservative*) NP-net modelling a bank customer service system. The SN is an environment where instances of two kind of agents interact, i.e., the elements of $\mathcal{N}$; we have agents α_1 and α_2 of type N_1 (customers), and an agent β_1 of type N_2 (employee). We could extract the elements of $\mathcal{N}$ from an event log, and to inject them as the initial marking of the net.

WF-nets consider an input and output place denoting the initial and final state of a process instance. Likewise, given a NP-net NP, we consider a set of input places $I = \{i_{N_1}, ..., i_{N_k}\}$ and output places $O = \{o_{N_1}, ..., o_{N_k}\}$, $I, O \subseteq P_{SN}$, and we consider the following rules: (i) in an initial marking M_0 of a NP, all net tokens of type N_i are assigned to a unique input place i_{N_i} s.t. $v(i_{N_i}) = N_i$, i.e., (see fig. 4) a customer is in its initial place waiting to be admitted, whereas an employee is in its initial place waiting for a customer to be served; (ii) in a final marking M_F of a NP, all net tokens of type N_i are expected to be located in an output place o_{N_i}, i.e., all customers are expected to be in their final place representing, that they left the bank; all employees are in their final place denoting their termination. With these conditions, we aim to formalize how a MAS process should be correct, i.e, to comply with some criterion of soundness for it. The following definition proposes such a criterion.

Definition 2. *Given a (strictly conservative) nested Petri net NP, with a stable set of net tokens $\mathcal{N}$, and with I and O as the set of input and output places, then NP is sound, iff:*

- *(Soundness of agents) Each WF-net (agent type) N_i is sound, according to the classical definition of soundness for WF-nets (see [4]).*
- *(Proper completion) Each net token $\alpha \in \mathcal{N}$ of type N_i is located in its assigned output place $o_{N_i} \in O$ in the final marking M_F.*
- *(Option to complete) From the initial marking M_0, it is always possible to reach the final marking M_F, i.e., each net token $\alpha \in \mathcal{N}$ should be able to arrive from its input place to its output place.*
- *(No dead transitions) For any $t \in T_{SN}$ there exists a possible firing sequence enabling t.*

We presented how to model MAS process models based on NP-nets, and how these should satisfy a criterion of soundness. Our ultimate goal, though, is the development of conformance heuristics, i.e., to compare a MAS process model against an event log to determine possible deviations of the observed behavior. Let us consider the following case:

$c_{NP} = \langle\ (t_1, \alpha_1), (a_1, \alpha_1), (b_1, \beta_1), (t_1, \alpha_1), (a_1, \alpha_2), (\lambda_1, \{\alpha_1, \beta_1\}), (a_3, \alpha_1),$
$(\lambda_3, \{\alpha_1, \beta_1\}), (\lambda_2, \alpha_2), (\lambda_4, \alpha_1), (\lambda_4, \alpha_2), (\lambda_5, \beta_1)\ \rangle$

The case c_{NP} is an ordered sequence of events related to the MAS process model of fig. 4. Each event is a pair $(a, \{\alpha_1, \alpha_2, ...\})$ indicating the occurrence of an activity $a \in A$, and a set of agents $\{\alpha_1, \alpha_2, ...\} \subseteq \mathcal{N}$ involved in such occurrence, i.e, (t_1, α_1) refers to activity t_1 executed in SN *transporting* α_1 (transport

step); (a_1, α_1) refers to a_1 executed by the agent α_1.(agent-autonomous step); and $(\lambda_1, \{\alpha_1, \beta_1\})$ refers to λ_1 executed by the agents α_1, β_1, and the SN (vertical synchronization step). Using a basic conformance checking technique, i.e, *token-based replay*, then c_{NP} perfectly fits in the model of fig. 4; if we count in the replay the number of produced (p), consumed (c), missing (m), and remaining (r) tokens on both, the agents and the system net, then the *fitness* metric $\frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p})$ will be equal to 1. Using an alignment technique, it is likely to expect the same result. However, considering non-fitting (deviating) cases, then we have to reason further on the analysis of deviations: deviation of some agents? of the system net? a deviation of an agent that affected other agents? This leads us to research on some conformance checking heuristic aware of multiple concurrent agents (within some environment) interacting.

4 Related Work

Conformance checking is increasingly attracting researchers and practitioners[3]. In the most recent book on conformance checking, Carmona et al. [8] classify conformance checking techniques in three types: (a) *rule-based checking*: to construct a set of rules, i.e, using *declare* models [21], to check whether cases comply these rules; (b) *token-replay*: to detect *missing* or *remaining* tokens along the replay of each event log trace in some model; and (c) *alignments*: to directly associate an event log trace with a valid execution sequence. Among these types, alignments provide the most advanced set of techniques. However, these techniques consider mostly the control-flow perspective, i.e, causal dependence between activities, thereby neglecting other causes in other perspectives that can be responsible in some case misbehavior; as a matter of fact, in [8] it is mentioned the need to develop techniques incorporating perspectives such as timestamps, resources, and in general, other case attributes (referred in [8] to as *multiple-perspective* conformance checking).

In this sense, it is noteworthy to cite the work by Mannhardt et al. [18], where it was developed an alignment-based multiple-perspective algorithm to detect case deviations using a cost function whose weight balances between the data attributes (time, resource, etc) and the control-flow; *Petri nets with data* were used. The approach of [18] is based on the work of de Leoni et al. [14], where the control-flow was fixed as the most important perspective in identifying deviations. In [5] it is proposed a method to verify whether cases in an event log meet a set of properties using a language based on Linear Temporal Logic (LTL). These rules may be defined over event attributes, so it can be computed whether some deviation of the case has occurred based on data. The later could be used in our work stating a set of rules that cases handling orders should comply.

[3] As a notorious example, the business-to-business firm *MarketAndMarkets* predicts conformance checking as the fastest growing segment of the process analytics market, that will worth USD 1.422 million by 2023 (see `https://www.marketsandmarkets.com/PressReleases/process-analytics` for more information).

Recent works address the importance to consider non-isolated cases. In the work of Denisov et al. [9] on performance analysis and process mining applied on logistics, it is considered the use of a *performance spectrum* that considers non-isolated cases, i.e, the time needed to serve a case may depend on other concurrent cases being served. In [10], D. Fahland studies the unaddressed challenges on process mining about processes interacting with each other in a one-to-many or many-to-many fashion; such interactions among processes may be suitably modelled by the use of NP-nets as we described in section 3.2.

Nowadays, the application of process mining has expanded to broad domains beyond the traditional business process management in organizations. Novel works has recently applied process mining on the performance analysis of railways [17], in the modelling of gaming behavior [22], and in the assessment of software development teams [7], among other examples. However, to the best of our knowledge, there have not been works applying process mining on trading systems. Our future work may bring new approaches based on process mining to validate trading systems that, for instance, the software testing industry may leverage within their testing activities on trading software.

5 Conclusions

In this paper we presented our research about the modelling and validation of trading and multi-agent systems. We presented our approach consisting in the phases of modelling, data pre-processing, simulation and conformance checking. We aim to use classes of Petri nets to model the expected behavior of processes in trading systems, as well as for constructing simulations, i.e, showing the dynamics of components like order books. We also generate event logs from the observed behavior of real systems, pre-processing samples of messages exchanged between users and the trading system. We envision that these event logs may be replayed in the simulation models. We also introduced how nested Petri nets may be used to model multi-agent system (MAS) processes, and we gave an idea about how to define correctness in MAS processes. Our ultimate research goal is to come up with data- and multiple-instance-aware conformance checking techniques. The use of simulation models may be helpful to reason on the desired functioning of the systems, and it may allow us to come up with new conformance checking heuristics.

Albeit we introduced separately the topics of trading and multi-agent systems, we indeed aim to combine the theory of MAS in the modelling and validation of trading systems. For instance, traders (or trading agents) play a determinant role in the state of order books, since they constantly interact with the trading system, i.e, submitting orders for buying and selling securities. Thus, a trading system with the traders may be modelled in a multi-agent structure. For example, inspired by the layer configuration of DB-nets [19], fig. 5 shows how a simulation model may be organized in our future research work to emulate traders (modelled as net tokens of a NP-net) interacting with the trading system (modelled within the system net of a NP-net) in order to apply some actions over order books (modelled as a database structure).

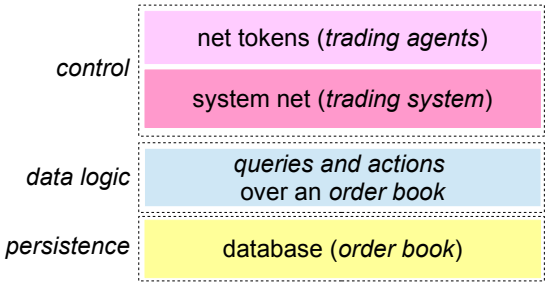

Fig. 5: Conceptual organization of a MAS model interacting with an order book. Inspired by the approach of DB-nets given in [19].

References

1. CPN Tools - A tool for editing, simulating, and analyzing Colored Petri nets. `https://www.cpntools.org`
2. FIX Standards - FIX Trading Community. `https://www.fixtrading.org/standards/`
3. London Stock Exchange - MIT 202 - FIX Trading Gateway Issue 11.8, 2018
4. van der Aalst, W.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers **08**(01), 21–66 (1998)
5. van der Aalst, W., de Beer, H., van Dongen, B.: Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In: Meersman, R., Tari, Z. (eds.) On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE. LNCS, vol. 3760, pp. 130–147. Springer (2005)
6. van der Aalst, W.: Process Mining: Data Science in Action. Springer, 2nd edn. (2016)
7. Caldeira, J., Abreu, F., Reis, J., Cardoso, J.: Assessing Software Development Teams Efficiency using Process Mining. In: 1st International Conference on Process Mining (ICPM) (2019)
8. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking: Relating Processes and Models. Springer (2018)
9. Denisov, V., Fahland, D., van der Aalst, W.: Unbiased, Fine-Grained Description of Processes Performance from Event Data. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) Business Process Management. LNCS, vol. 11080, pp. 139–157. Springer (2018)
10. Fahland, D.: Describing Behavior of Processes with Many-to-Many Interactions. In: Donatelli, S., Haar, S. (eds.) Application and Theory of Petri Nets and Concurrency. LNCS, vol. 11522, pp. 3–24. Springer (2019)
11. Harris, L.: Trading and Exchanges: Market Microstructure for Practitioners. Oxford University Press (2003)
12. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer (2009)
13. Kummer, O., Wienberg, F., Duvigneau, M., Schumacher, J., Köhler, M., Moldt, D., Rölke, H., Valk, R.: An Extensible Editor and Simulation Engine for Petri Nets: Renew. In: Cortadella, J., Reisig, W. (eds.) Applications and Theory of Petri Nets 2004. LNCS, vol. 11080, pp. 484–493. Springer (2004)

14. de Leoni, M., van der Aalst, W.: Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management. LNCS, vol. 8094, pp. 113–129. Springer (2013)
15. Lomazova, I.A.: Nested Petri Nets - a Formalism for Specification and Verification of Multi-Agent Distributed Systems. Fundamenta Informaticae **43**, 195–214 (2000)
16. Lomazova, I.A., Ermakova, V.O.: Verification of Nested Petri Nets Using an Unfolding Approach. In: Cabac, L., Kristensen, L.M., Rölke, H. (eds.) Petri Nets and Software Engineering. CEUR Workshop Proceedings, vol. 1591 (2016)
17. Mannhardt, F., Arnesen, P., Landmark, A.: Estimating the Impact of Incidents on Process Delay. In: 1st International Conference on Process Mining (ICPM) (2019)
18. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016)
19. Montali, M., Rivkin, A.: DB-Nets: On the Marriage of Colored Petri Nets and Relational Databases". In: Koutny, M., Kleijn, J., Penczek, W. (eds.) Transactions on Petri Nets and Other Models of Concurrency XII. LNCS, vol. 10470, pp. 91–118. Springer (2017)
20. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
21. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: Full Support for Loosely-Structured Processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). pp. 287–287 (2007)
22. Ramadan, S., Baqapuri, H., Roecher, E., Mathiak, K.: Process Mining of Logged Gaming Behavior during Functional Magnetic Resonance Imaging. In: 1st International Conference on Process Mining (ICPM) (2019)