1

Process Mining in a Multi-Agent Auctioning System

B.F. van Dongen, J. van Luin, and H.M.W. Verbeek

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{b.f.v.dongen,j.v.luin,h.m.w.verbeek}@tm.tue.nl

Summary. Both *process mining* and *multi agent simulations* are relatively new research areas, both evolving rapidly. In multi agent simulations, techniques are being developed to make agents more intelligent and more adaptive to their environment. In process mining, the focus has shifted from discovering a complete process model from an execution log, to the development of all kind of analysis techniques related to processes. In this paper, we combine the two research areas by showing that process mining techniques, applied to the communication logs of multi-agent simulations, can help in understanding the behaviour of agents. Furthermore, we show that process mining could actually help agents to adapt to their environment, by enabling them to discover *why* another agent makes certain decisions instead of just reacting on the fact *that* the other agent made a decision.

Key words: Process Mining, Multi-Agent Simulation, Auctioning Systems

1.1 Introduction

Over last few years, the focus of *process mining* research has shifted. In [3], the research field was still called "workflow mining" and the focus was on how to derive a complete process model from an execution log of a real life process (nowadays called *control flow rediscovery*). Although new algorithms for deriving process models from execution logs are still being developed, more and more attention is given to other means of analyzing process logs, often inspired by real life applications, or developments in legislation. However, since we are convinced that we should not look at all this research in isolation. In fact, the power of process mining is in the integration of many different analysis methods. Therefore, in this paper, we show how several aspects of process mining, can be combined to give great insight in ongoing processes.

For this purpose, we investigate a so-called *multi-agent auctioning system*. Such a system is a complex simulation of a real life auction, i.e. one seller is selling goods trough an auctioneer and many buyers are buying these goods

from that auctioneer by placing bids. The reason for looking at such an agent system to proof that process mining techniques should be combined is simple. Participants in auctions have conflicting goals: the seller wants to receive a high price for his goods, and the buyers want to pay a low price. In repeated multi-unit uniform-price auctions (see section 1.2.1), analysis of past behaviour of the participants can be used by the seller as well as the buyers to improve their results. In this paper, we show how process-mining of auction event-logs can be used to analyze the behaviour, and derive models of the competing agents, and how this analysis can be used to get better results.

1.2 Multi-Agent Simulation

We consider a multi-agent game where agents represent factories in a supply chain. At the start of a round, customers place their orders with the factories. Each factory has until the end of the round to produce the products to satisfy their demand. Orders that are not filled at the end of a round are lost. The factories can purchase the raw material they need for production in an auction that takes place directly after the customer demand is known. When the auction is over, the materials are delivered to the winning agents immediately and each agent can start its production. For simplicity reasons, factories have an unlimited inventory and an unlimited production capacity. The flow of money and goods in our agent system is presented in Figure 1.1.

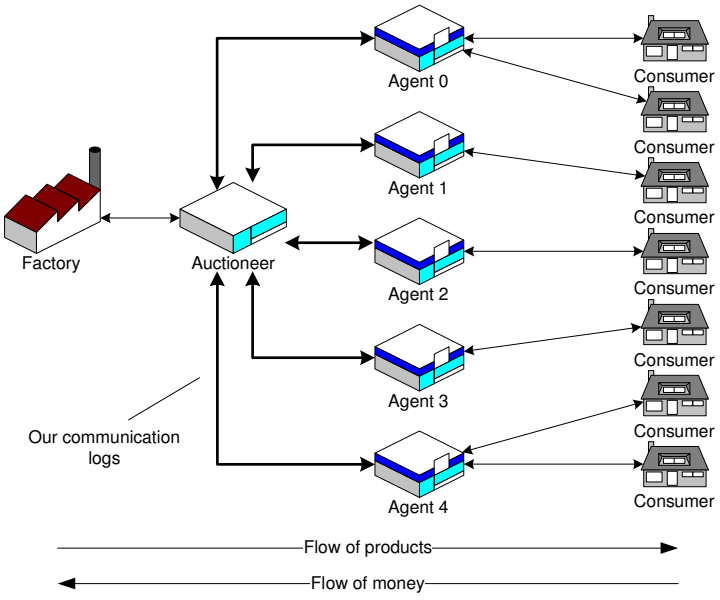


Fig. 1.1. The flow of money and goods in our agent system.

1.2.1 Auction

The auctions that we use are uniform-price multi-unit auctions [11]. In this type of auctions, a large number of identical products are sold, and each agent can bid on a share of these products, by submitting a bid containing the desired amount of products and the price per unit he is willing to pay. Bids with an amount of 0 or with negative amounts are removed. The agent with the highest price-per-unit in his offer wins the amount of products he requested. If the amount of products for sale was higher than the amount won by the highest bidder, the second-highest bidder wins either his share, or the remaining amount of products, whichever is less. This process is repeated until either all participants received their share, or no products are left. When the last allocated share contains less products than that agent bid for, the agent has to accept the lower amount.

The price per unit that is payed by the agents that received a share is equal to the highest price offered by the agents that did not receive anything. In the case that all agents received some amount of products, the reserved price is payed by all.

1.2.2 Inventory management

Each agent uses a base-stock inventory management strategy[16]. In this strategy, the agent defines a reorder-point: the amount of raw material the agent wants to have in stock to compensate for the uncertainty of winning in an auction. Each round, the agent will request an amount of raw material that is high enough to produce the customer demand for that round, and bring the inventory to the reorder-point level:

$$(requested) = (reorder - point) + (current\ demand) - (current\ inventory)$$

The inventory management strategy of each agent is known to the agent only, i.e. the auctioneer has now idea about the inventory management of an agent. However, after introducing the area of process mining in Section 1.3, we show in Section 1.4 that even without this knowledge, the auctioneer is able to get some insight into *why* agents place a bid to replenish their inventories.

1.3 Process Mining

The research domain *process mining* is relatively new. A complete overview of recent process mining research is beyond the scope of this paper. Therefore, we limit ourselves to a brief introduction to this topic and refer to [3, 4] and the http://www.processmining.org web page for a more complete overview.

The goal of process mining is to extract information about processes from transaction logs. It assumes that it is possible to record events such that (i)

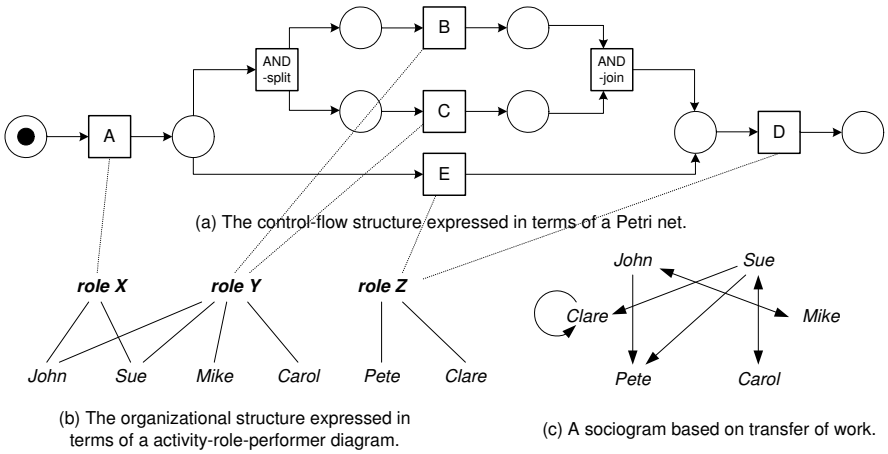
case id	activity id	originator	case id	activity id	originator
case 1	activity A	John	case 5	activity A	Sue
case 2	activity A	John	case 4	activity C	Carol
case 3	activity A	Sue	case 1	activity D	Pete
case 3	activity B	Carol	case 3	activity C	Sue
case 1	activity B	Mike	case 3	activity D	Pete
case 1	activity C	John	case 4	activity B	Sue
case 2	activity C	Mike	case 5	activity E	Clare
case 4	activity A	Sue	case 5	activity D	Clare
case 2	activity B	John	case 4	activity D	Pete
case 2	activity D	Pete			

Table 1.1. An event log (audit trail).

each event refers to an *activity* (i.e., a well-defined step in the process), (ii) each event refers to a *case* (i.e., a process instance), (iii) each event can have a *performer* also referred to as *originator* (the actor executing or initiating the activity), and (iv) events have a *timestamp* and are totally ordered. Table 1.1 shows an example of a log involving 19 events, 5 activities, and 6 originators. In addition to the information shown in this table, some event logs contain more information on the case itself, i.e., data elements referring to properties of the case.

Event logs such as the one shown in Table 1.1 are used as the starting point for mining. We distinguish three different perspectives: (1) the process perspective, (2) the organizational perspective and (3) the case perspective. The *process perspective* focuses on the control-flow, i.e., the ordering of activities, which is shown in terms of a Petri net (cf. [14]) in Figure 1.2(a). The goal of mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net or Event-driven Process Chain (EPC) [10, 12]. The *organizational perspective* focuses on the originator field, i.e., which performers are involved and how are they related. The goal is to either structure the organization by classifying people in terms of roles and organizational units (Figure 1.2(b)) or to show relation between individual performers (i.e., build a social network as described in [2] and references there, and as shown in Figure 1.2(c)). The *case perspective* focuses on properties of cases. Cases can be characterized by their path in the process or by the originators working on a case. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order, it is interesting to know the supplier or the number of products ordered.

Orthogonal to the three perspectives (process, organization, and case), the result of a mining effort may refer to *logical* issues and/or *performance* issues. For example, process mining can focus on performance issues such as flow time, the utilization of performers or execution frequencies.



(a) The control-flow structure expressed in terms of a Petri net.

(b) The organizational structure expressed in terms of a activity-role-performer diagram.

(c) A sociogram based on transfer of work.

Fig. 1.2. Some mining results for the process perspective (a) and organizational (b and c) perspective based on the event log shown in Table 1.1.

In this paper, we use three aspects of process mining in particular. First of all, we try to generate the control flow of a process from a log. We do not specify a specific algorithm that we use for this. Instead, most of the algorithms implemented in the ProM framework, which will be introduced later, are able to correctly discover the control flows that we are looking for. Furthermore, we use a concept called "decision mining". In decision mining, a control flow is analyzed and for each choice that has to be made, data is used to predict which choice will be made in future cases. Finally, we analyse process logs using a linear temporal logic, as described in [1]. Questions, such as how often did an agent not deliver its customer order are answered with that logic, which, just like all other algorithms, is implemented as a plug-in in the ProM framework.

1.3.1 The ProM Framework

The (Pro)cess (M)ining framework *ProM* has been developed as a completely plug-able environment for process mining and related topics. It can be extended by simply adding plug-ins, i.e., there is no need to know or to recompile the source code. Currently, more than 80 plug-ins have been added. The ProM framework has been described before in [9, 17] and of course the web site www.processmining.org. The most interesting plug-ins in the context of this paper are the mining plug-ins and the analysis plug-ins. The architecture of ProM however allows for five different types of plug-ins:

Mining plug-ins which implement some mining algorithm, e.g., mining algorithms that construct a Petri net based on some event log.

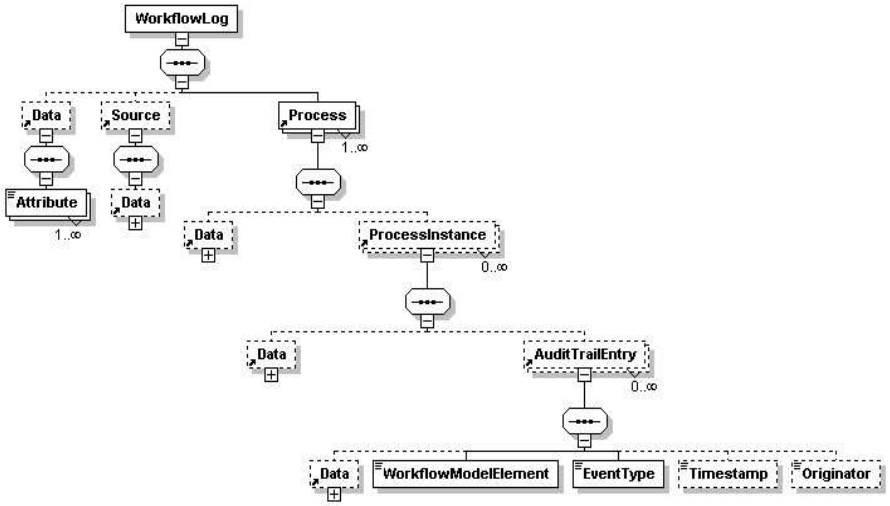


Fig. 1.3. MXML schema Definition.

Export plug-ins which implement some "save as" functionality for some objects (such as graphs). For example, there are plug-ins to save EPCs, Petri nets, spreadsheets, etc.

Import plug-ins which implement an "open" functionality for exported objects, e.g., load instance-EPCs from ARIS PPM.

Analysis plug-ins which typically implement some property analysis on some mining result. For example, for Petri nets there is a plug-in which constructs place invariants, transition invariants, and a coverability graph.

Conversion plug-ins which implement conversions between different data formats, e.g., from EPCs to Petri nets.

In Table 1.1 we presented a fragment of an event log. We assume a standard log format, named MXML as described in [7], and have developed several adaptors to map logs in different information systems onto our log format (e.g., Staffware, FLOWer, MS Exchange, MQSeries, etc.). Figure 1.3 shows the hierarchical structure of MXML. The format is XML based and is defined by an XML schema (cf. www.processmining.org).

1.4 Process Mining in Multi Agent Simulation

Process mining can be relevant in multi agent simulations for several reasons. First, process mining can be used to test hypotheses on the behaviour of an agent by inspecting the agent's communication logs. Furthermore, as people do in real life, we would like our agents to be able to determine the behaviour of its competitors and react on that. Agents are pieces of software with one

or more behaviours and one or more goals. Although the parameters of the behaviours and goals often change over time, the internal structure is likely to stay the same. Therefore, understanding the structure of an agent (its process model) can help the buyers as well as the auctioneer to anticipate the behaviour of that agent.

In this section, we consider some process mining techniques that can be used in agent simulations. First, we show how communication logs can be translated to event logs. Then we show how the behaviour of an agent can be analyzed using these event logs, for example to test hypotheses on their behaviour. We then show how process mining can be used to determine the internal processes of the agent. Finally, we show the same process logs can then be used to find the way decisions are made by the agent.

1.4.1 Translating agent logs to process logs

In multi agent systems, agents interact with each other by sending messages. Therefore, the communication log of such a system can be seen as one big message sequence chart. In our case, we will focus on the messages passed to and from the auctioneer, since these are the messages that the auctioneer has access to. We know that each message going from the auctioneer to one of the agents requires an action by the agent and it is safe to assume that a real life auctioneer would also know this. The auctioneer also knows that each agent receives an order from a customer at the beginning of a round (although the customer can order 0 products) and that the agent will deliver goods to its customer if possible at the end of the round. Furthermore, the auctioneer knows that if the agent places a bid for 0 products in the auction, it apparently had enough products to supply its customers. Finally, the auctioneer knows that each agent can continuously change the price it offers for the goods. Figure 1.4, shows a sequence of messages passed between the auctioneer and one agent for one auction.

The message sequence chart in Figure 1.4 and the textual description thereof is the only knowledge that we assume the auctioneer possesses. However, the auctioneer wants to gain more insight into the behaviour of each

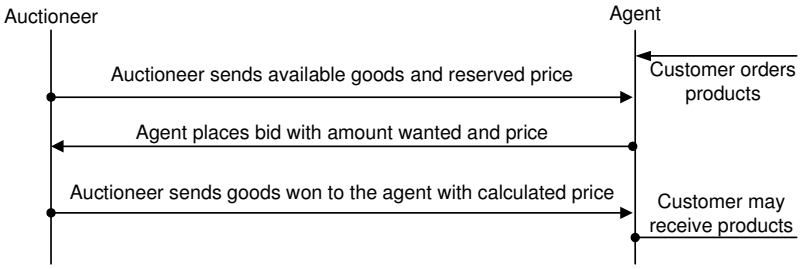


Fig. 1.4. Sequence of messages between auctioneer and agent for 1 auction.

agent that it communicates with and for this purpose, techniques from the domain of process mining are used. The first step the auctioneer has to take in order to gain knowledge is to translate the communication logs of the auctioneer with an agent to MXML.

The agent first receives an order from its customers "Receive Order" in the beginning of the month, after which the agent places a bid at the auctioneer "Place Bid", or doesn't place a bid at the auctioneer "Don't Place Bid". However, before placing a bid, the agent determines whether or not to change the price it offers, i.e. through activities "Increase Price", "Decrease Price", "Keep Price". Then the auctioneer distributes the products that are received by the agent "Receive goods". Finally, the agent either delivers the goods to its customer or does not deliver them, which cannot be seen by the auctioneer and is therefore not included in the MXML log.

The translation above describes all the knowledge of the auctioneer with respect to the *behaviour* of the agents. However, the auctioneer also has access to some *data* of the agent, namely the data that is exchanged in the messages. In our case, this means that the auctioneer has access to the following data:

- How many products did the agent order and/or receive at the last auction?
- What were the offered and/or paid prices for the last auction?
- How much products does the agent order and/or receive this auction?
- What price does the agent offer and/or pay this auction?

These data fields are included in the process log as well. The essence of the translation of this sequence of messages in MXML (from the viewpoint of the auctioneer) is given in Figure 1.5.

In Figure 1.6, we show a screenshot of ProM, after opening the MXML log of agent 0, from the perspective of the auctioneer. It clearly shows the same information as Figure 1.5.

In the remainder of this section, we first show what the auctioneer can do with the process log it generated. After that, we show what the agent himself could do with a similar process logs.

1.4.2 Mining the agents behaviour

At this point, we have translated the communication log of the auctioneer with one agent to an event log for use in process mining. We assume that we have a log from the viewpoint of the auctioneer, i.e. a log containing some 1000 process instances with the data described in Section 1.4.1. The first focus of the auctioneer is to figure out the behaviour of an agent by determining its process model. This may seem strange, since we explicitly introduced some sort of a process when the logs were translated. However, an agent can restrict the behaviour of this simple model and it may be worthwhile to investigate this in more detail. At this point, the auctioneer uses a process mining algorithm, tailored towards control flow rediscovery. In fact, many of these algorithms exist today and most of them are implemented in the ProM framework. As an

```
<ProcessInstance id="auction 18">
    <Data>
      <Attribute name="last order">16</Attribute>
      <Attribute name="last won">0</Attribute>
      <Attribute name="last price">0</Attribute>
      <Attribute name="last bid">11</Attribute>
    </Data>
  <AuditTrailEntry>
    <WorkflowModelElement>Receive order</WorkflowModelElement>
  </AuditTrailEntry>
  <AuditTrailEntry>
    <WorkflowModelElement>Increase Price</WorkflowModelElement>
  </AuditTrailEntry>
  <AuditTrailEntry>
    <WorkflowModelElement>Place Bid</WorkflowModelElement>
    <Data>
      <Attribute name="amount">26</Attribute>
      <Attribute name="price">12</Attribute>
    </Data>
  </AuditTrailEntry>
  <AuditTrailEntry>
    <WorkflowModelElement>Receive Goods</WorkflowModelElement>
    <Data>
      <Attribute name="received amount">0</Attribute>
      <Attribute name="received price">0</Attribute>
      <Attribute name="ordered">true</Attribute>
    </Data>
  </AuditTrailEntry>
</ProcessInstance>
```

Fig. 1.5. Figure 1.4 translated to MXML by the auctioneer.

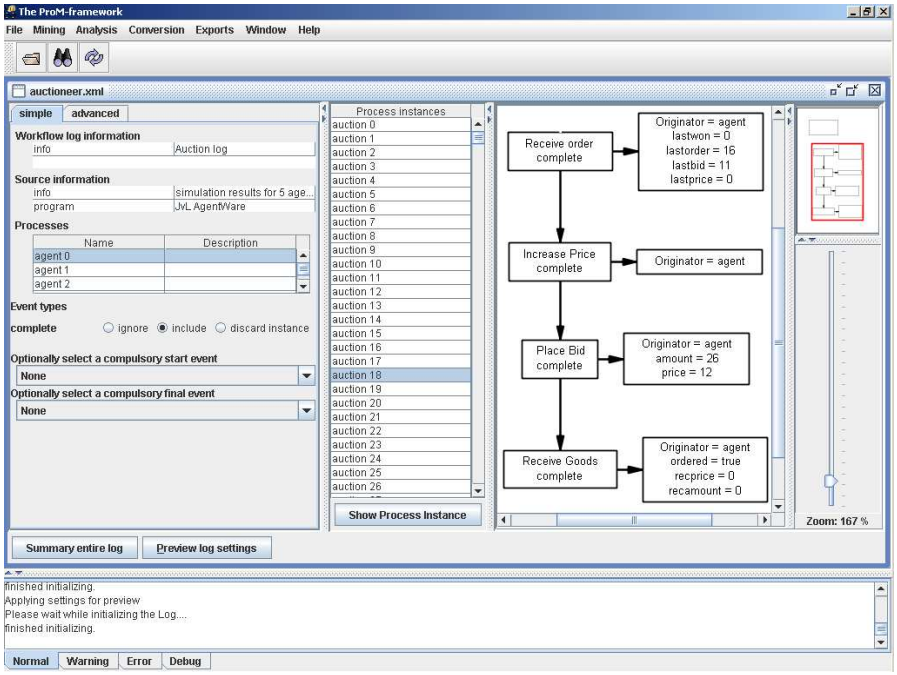


Fig. 1.6. Screenshot of ProM showing the log of Figure 1.5.

example, we used the multi-phase mining plugin. . The basic idea behind this algorithm is simple. First, based on the entire set of 1000 process instances, causal dependencies are derived between activities in the following way. If one activity A is directly followed by B at least once and A is never directly preceded by B then there is a causal dependency from A to B. In the second step, each process instance is translated into a partial order instead of a linear order on its audit trail entries [6]. Finally, in the last step, these partial orders are aggregated and the types of split and joins are derived [8].

Figures 1.7 and 1.8 were generated by ProM using the multi-phase mining plugin. Actually, many algorithms give the same result, such as [5] and the heuristics miner, which is a ProM implementation of [18]. Both of these algorithm also first determine causal dependencies between activities, but they use heuristics to do so and are therefore more robust with respect to logs that contain "noise", i.e. logs where activities appear in the wrong order, or do not appear at all, for example if a communication between the auctioneer and an agent breaks down.

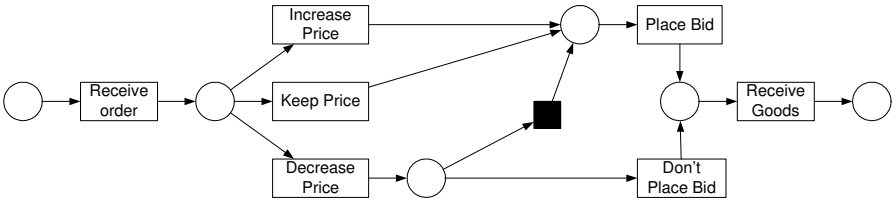


Fig. 1.7. Agent 0 as perceived by the auctioneer.

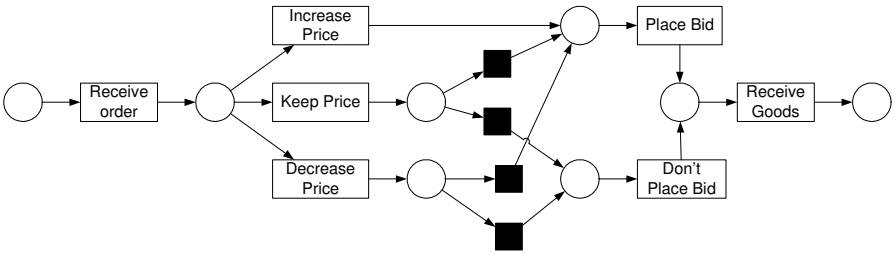


Fig. 1.8. Agent 1 as perceived by the auctioneer.

Besides the activities that we described before, both figures show some internal actions without a label. The algorithms in ProM insert these steps automatically, to capture the behaviour of the process as good as possible. Compared to the process described earlier, both agents do not seem to show all the possible behaviour. For example, agent 0 chooses not to place a bid

after he decreased his price, furthermore, both agents always place a bid if they increase their price. It is not hard to see that formally it is possible for the situation to occur where the price is increased and no bid is placed. However, in our logs, it does not happen.

In an auction, the seller wants to earn as much money as possible, whereas the buyers want to buy at the lowest possible price. In earlier research [13], we saw that intelligent (adaptive) buyers can learn to use the uniform-price property of the auction to their advantage. The agents learned to accept an even-split of the raw materials, and made sure that every agent received some raw material. Because of the uniform-price property, every buyer payed the lowest possible price: the reserved price. For the seller (in our simulations represented by the auctioneer), this is a undesirable result.

To counter this behaviour, the auctioneer needs to get a better insight into the decisions made by the buyers. One of the instruments that the auctioneer can use is the reserved-price. If the auctioneer can determine the decisions made by the buyers to calculate the price they would be willing to pay, he can find a better value for the reserved-price. In process mining, this is called *decision mining*. As an example, we will try to find out *why* an agent changes its price.

It is important to realize that the result of control flow mining is not always a perfect process specification. However, as agents are pieces of software that have an internal logic given by its programming code, we can safely assume that we can find a reasonable Petri net. Furthermore, the analysis methods presented in the remainder of this paper are robust enough to deal with Petri nets that do not conform to the actual behaviour for 100 percent.

1.4.3 Analyzing decision points

As we saw in figures 1.7 and 1.8, the auctioneer is capable of constructing the internal process within an agent. Using this model of an agent and the process log for an agent, a so-called decision point analysis can be performed. In ProM, this analysis is implemented in an analysis plugin called the "Decision Miner". The Decision Miner analyzes how data attributes influence the choices made in a process based on the given process instances. More precisely, it aims at the detection of data dependencies that affect the routing of a process instance. The approach is based on machine learning techniques, and the Decision Miner uses algorithms implemented in the Weka [19] library to solve the classification problem. For details on the algorithms used, we refer to [15].

In our case, the auctioneer has to figure out if the choice between "Increase Price", "Decrease Price" or "Keep Price" is made arbitrarily by the agent, or whether it was made based on data that is also available to the auctioneer. As can be seen in the process model of figure 1.7 and 1.8, the auctioneer assumes that the agent determines its new price only at the beginning of each auction. Using the MXML log files, the Decision Miner knows that only the four data elements above are available at that point in time. Therefore, in the

ProM framework, we applied the decision point analysis to the model of agent 1, where we included all the numerical attributes available to the auctioneer that are present at the time the decision has to be made. These attributes are:

last won The number of products the agent won in the last auction,
last order The number of products ordered in the last auction,
last bid The price offered in the last auction,
last price The price paid in the last auction.

Figure 1.9 shows the decision tree for updating the price for Agent 1[1]. This decision tree shows that if "last won" equals 0 and "last order" is more than 0, the price is increased. The decision tree is somewhat large (i.e. it is overfitting the data), however it correctly classifies all the 1000 auctions. Furthermore, if the algorithm is applied to the data of all the agents, (i.e. 5000 auctions) the decision tree is more compact (see Figure 1.10) and it again classifies all auctions correctly.

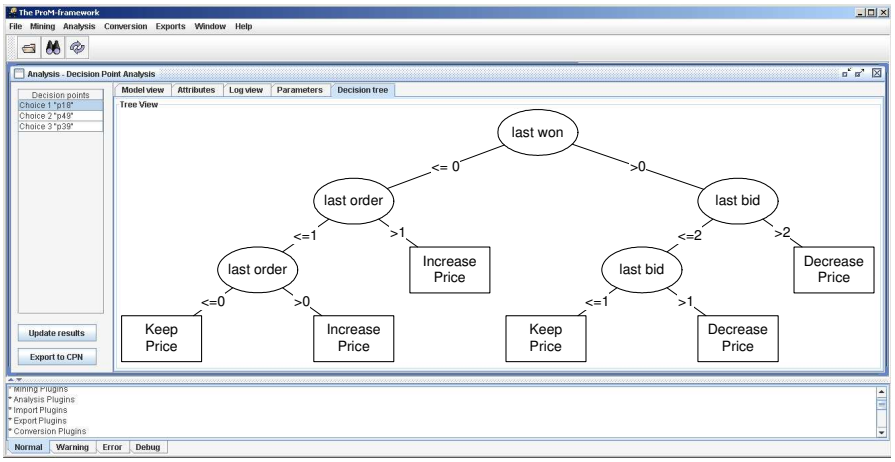


Fig. 1.9. Decision tree for updating the price by Agent 1.

At this point, the auctioneer would have enough information to "cheat" its auctioning system, since it seems that the way to avoid agents to decrease their price is by not giving them any products. Of course, this is not a practical scenario. However, the auctioneer could decide in every auction to have at least one agent that did place a bid, but does not get any goods (the auctioneer could throw them away). In the next round this agent will raise its price and this way, the auctioneer can keep the price at an artificially high level.

[1] Note that the screenshot is touched-up for readability. The ProM framework however, shows a graph containing the same information.

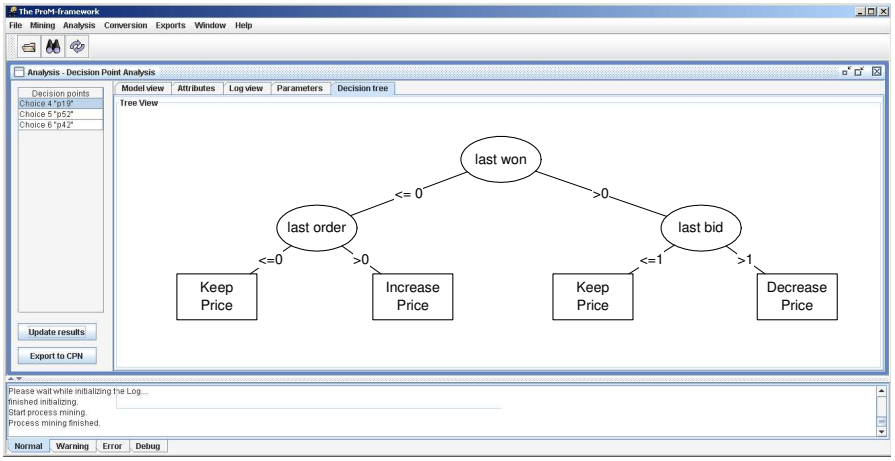


Fig. 1.10. Decision tree for updating the price by all agents.

1.4.4 Agent self-assessment

So far, we focussed on the the event log from the viewpoint of the auctioneer. The same log from the viewpoint of the agent however is more extensive, since it contains information about the inventory level of the agent, the current re-order point and whether or not goods were delivered to the customer. Furthermore, the agent itself has clear knowledge of its process model, so there is no need for him to discover it. Moreover, in practice it is not always possible to discover a correct process model from a process log. However in this section, we show that process mining can also be useful in other ways.

Although there is no need for an agent to discover its own process model, we are still interested how an agent could improve itself by looking at its internal process. For this, we generated a more extensive log that also includes the internal data of each agent. As we know from the introduction, each agent tries to keep its costs as low as possible, but also it tries to keep its customers satisfied. This gives rise to two questions:

1. How often is an agent not capable of delivering to its customer?
2. How often is an agent capable of delivering to its customer without having to place a bid at the auction?

The first question relates directly to customer satisfaction. A customer is satisfied if his goods are delivered and he is not satisfied if this is not the case. The second question needs a bit more insight. If an agent can deliver to its customer without having to place a bid at the auction, it implies that its current inventory is so high that it is greater than the re-order point plus the customer demand, implying that the agent ordered too many goods in a previous auction, which raises the inventory costs.

```
formula Cannot_Deliver( ) :=
  {Check if the agent could deliver or not, i.e. eventually
   activity "Dont deliver" is executed.}
      <>( ate.WorkflowModelElement == "Dont deliver" );

formula Deliver_Without_Bid( ) :=
  {Check if the agent delivered but didn't order in the same
   auction, i.e. eventually the activity "didnt order" shows
   up in the log, and also the activity "Deliver" does.}
    ( <>( ate.WorkflowModelElement == "Deliver" ) /\
      <>( ate.WorkflowModelElement == "didnt order" ) );
```

Fig. 1.11. LTL Formulas for self-assessment of an agent.

Again, we use process mining techniques to answer these two questions. However, there is an important difference with the previous approaches, since we can now assume that we have complete knowledge of the actual process model describing the agent. Therefore, constructing an MXML process log from the viewpoint of an agent is more easy since there is no need to construct them from a message sequence chart. In fact, we can just use the execution log of an agent directly (provided that it is given in the MXML format)

The way to answer the two questions in the context of process mining is by using Linear Temporal Logic (LTL) on the process log. In ProM this is implemented as an analysis plug-in called the LTL-checker [1], that can check logical expressions that involve time on a log. Using this plug-in, we check the two questions posted above using the LTL file of Figure 1.11 containing two expressions.

LTL formulas are logical formulas using a temporal logic. The formulas presented in Figure 1.11 should be read as follows:

- The formula "Cannot_Deliver" checks whether *eventually* (denoted by `<>`) the workflow model element in an audit trail entry (see the definition of MXML) *equals* (denoted by `==`) "Dont deliver". In other words, does the activity "Dont deliver" appear anywhere in the process instance.
- The formula "Deliver_Without_Bid" checks whether *eventually* the workflow model element in an audit trail entry *equals* "Deliver", *and* (denoted by `/\`) *eventually* the workflow model element in an audit trail entry *equals* "didnt order" In other words, do both activities "Deliver" and "didnt order" appear anywhere in the process instance.

We checked those two formulas on the logs generated by all five agents and the results are presented in Table 1.2. From this table, it is clear that the five agents are all quite good in keeping a low inventory level, since only in 1 to 2% of the auctions they can deliver goods to their customers without placing a bid. However, with the exception of agent 2, around 40% of the customer

Agent	Total auctions	Unable to deliver	Delivered without ordering
Agent 0	1000	443	11
Agent 1	1000	416	13
Agent 2	1000	238	26
Agent 3	1000	466	17
Agent 4	1000	325	11

Table 1.2. Results for self-assessment of the agents.

orders are not delivered at all, which is a disturbing result. In a real situation, not being able to deliver 40% of the orders is unacceptable.

1.5 Conclusion

In this paper, we presented a way how different aspects of process mining could be used in multi agent systems. We have shown that some of the aspects of process mining, i.e. "control flow rediscovery", "decision mining" and "LTL verification", can be useful to enhance the capabilities of a multi agent simulation system in different ways.

Primarily, for one agent to be able to discover the internal structure of another agent can help him in understanding and predicting its competitors behaviour. Instead of observing *what* decisions other agents are making, he can tell *why* they do so and react on that. Therefore, using a combination of control flow rediscovery and decision mining, an agent could in theory at least, adapt to its environment in a way we humans do.

Furthermore, using techniques such as the verification of logical properties, we are able to understand the behaviour of our agents better and using that understanding we are able to tweak the simulation settings to get results that are more consistent with real life situations.

References

1. W.M.P. van der Aalst, H.T. de Beer, and B.F. van Dongen. Process Mining and Verification of Properties: An Approach based on Temporal Logic. In *OTM Conferences*, volume 3761 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, Berlin, 2005.
2. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, Berlin, 2004.
3. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

4. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
5. A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic Process Mining: A Basic Approach and its Challenges. In *BPM - BPI workshop*, 2005.
6. B.F. van Dongen and W.M.P. van der Aalst. Multi-phase Process mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *Conceptual Modeling - ER 2004*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.
7. B.F. van Dongen and W.M.P. van der Aalst. A Meta Model for Process Mining Data. In *Proceedings of the CAiSE WORKSHOPS*, volume 2, pages 309–320. FEUP, 2005.
8. B.F. van Dongen and W.M.P. van der Aalst. Multi-phase Process mining: Aggregating Instance Graphs into EPCs and Petri Nets. In *PNCWB 2005 workshop*, pages 35–58, 2005.
9. B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
10. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.
11. J.H. Kagel and D. Levin. Behavior in multi-unit demand auctions: experiments with uniform price and dynamic vickrey auctions. *Econometrica*, 69(2):413–454, March 2001.
12. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
13. J. van Luin, J.A. La Poutré, and J.W.M. Bertrand. Learning inventory management strategies for commodity supply chains with customer satisfaction. In *Proceedings of the Eighth International Conference on Electronic Commerce (ICEC06)*, 2006 (to appear).
14. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
15. A. Rozinat and W.M.P. van der Aalst. Decision mining in Business Processes. BPM Center Report BPM-06-10, 2006.
16. E.A. Silver, D.F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John Wiley & Sons, 3rd edition, 1998.
17. H.M.W. Verbeek, B.F. van Dongen, J. Mendling, and W.M.P. van der Aalst. Interoperability in the ProM Framework. In *Accepted at EMOI 2006*, 2006.
18. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
19. I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.