Improving Business Process Models with Agent-Based Simulation and Process Mining

Fernando Szimanski[1], Célia G. Ralha[1], Gerd Wagner[2], and Diogo R. Ferreira[3]

[1] University of Brasília, Brazil
fszimanski@gmail.com, ghedini@cic.unb.br
[2] Brandenburg University of Technology, Cottbus, Germany
gwagner@informatik.tu-cottbus.de
[3] IST – Technical University of Lisbon, Portugal
diogo.ferreira@ist.utl.pt

Abstract. Business processes are usually modeled at a high level of abstraction, while the analysis of their run-time behavior through process mining techniques is based on low-level events recorded in an event log. In this scenario, it is difficult to discover the relationship between the process model and the run-time behavior, and to check whether the model is actually a good representation for that behavior. In this work, we introduce an approach that is able to capture such relationship in a hierarchical model. In addition, through a combination of process mining and agent-based simulation, the approach supports the improvement of the process model so that it becomes a better representation for the behavior of agents in the process. For this purpose, the model is evaluated based on a set of metrics. We illustrate the approach in an application scenario involving a purchase process.

1 Introduction

Business processes are usually defined at a high level of abstraction using modeling languages such as BPMN [1], EPCs [2], and Petri nets [3]. In these types of models, the process is depicted as a sequence of activities, where each activity is to be performed by some agent. In human interaction workflows [4], the agent is typically a user who is able to perform certain tasks over a supporting systems infrastructure. During execution, when an agent is assigned to a certain activity, it performs a set of operations over the systems infrastructure. Among other things, such as data and information manipulation, these operations may include communicating with other agents as well.

When agents perform operations over a systems infrastructure, it is possible to record these actions in the form of events. Typically, each event refers to an operation that was performed by some agent during the execution of a process instance. Such events are recorded in an event log and, from this event log, it is possible to analyze the run-time behavior of agents through process mining techniques [5]. These techniques allow studying the run-time behavior from a number of perspectives, including the sequence of operations as well as the interactions that take place between agents during process execution.

The ultimate purpose of such analysis is to be able to compare the predefined behavior for the process with the actual behavior of agents at run-time. However, such goal faces a major obstacle. While business processes are defined and modeled at a high level, the analysis of run-time behavior through process mining techniques is based on the low-level events recorded in the event log, as each agent carries out its own operations. Clearly, there is a gap between the high level of abstraction at which processes are defined, and the low-level nature of events recorded in the event log.

The goal of this work is to bridge this gap and to provide a platform for the improvement of process models based on a combination of agent-based simulation [6–8] and process mining. Through agent-based simulation, it is possible to define, implement, and simulate a business process as a sequence of activities carried out by multiple agents working together. The interactions between these agents are recorded as low-level events. Through process mining, it is possible to extract models of behavior from this event log. However, the extraction must be done in such a way that it is possible to map low-level events to the high-level activities defined in the business process.

In summary, this work provides the following contributions:

– It shows that an agent-based simulation framework – specifically, the Agent-Object Relationship (AOR) framework [9] – can be used as a platform for implementing and simulating business processes.
– It describes a process mining technique that is able to extract a hierarchical Markov model [10] from an event log and from a description of the high-level process provided as input.
– It provides a set of metrics to evaluate the complexity of models obtained with such technique, which can serve as a guide when considering different ways to model the business process.
– It shows that the combination of both tools – i.e. the agent-based simulation framework and the proposed process mining technique – can be used as a testbed for trying out different models of the business process.

Section 2 provides an overview of the proposed approach. Section 3 discusses agent-based simulation, and in particular the use of the AOR framework in this work. Section 4 describes the hierarchical Markov model that is used to capture the run-time behavior of agents, and the algorithm to extract such model from an event log. Section 5 discusses metrics and the evaluation of the extracted models. Section 6 presents an application scenario involving in a purchase process. Finally, Section 7 concludes the paper.

2 Approach Overview

One of the premises for this work is that business experts will be able to provide a high-level description of the business process. Typically, the process is described in terms of a process model with a set of high-level activities. On the other hand, there are process mining techniques to extract the behavior of a business process from event logs, but the low-level events that are recorded in the event log may not have a clear relationship to the high-level activities defined in the process model. Therefore, one of the main issues to be addressed is how to map the low-level events recorded in an event log to the high-level activities defined in a process model. In our approach, this is done with the aid of a process mining technique which is able to extract a hierarchical Markov model from the event log and from a Markov-model representation of the high-level business process, as shown in Figure 1.

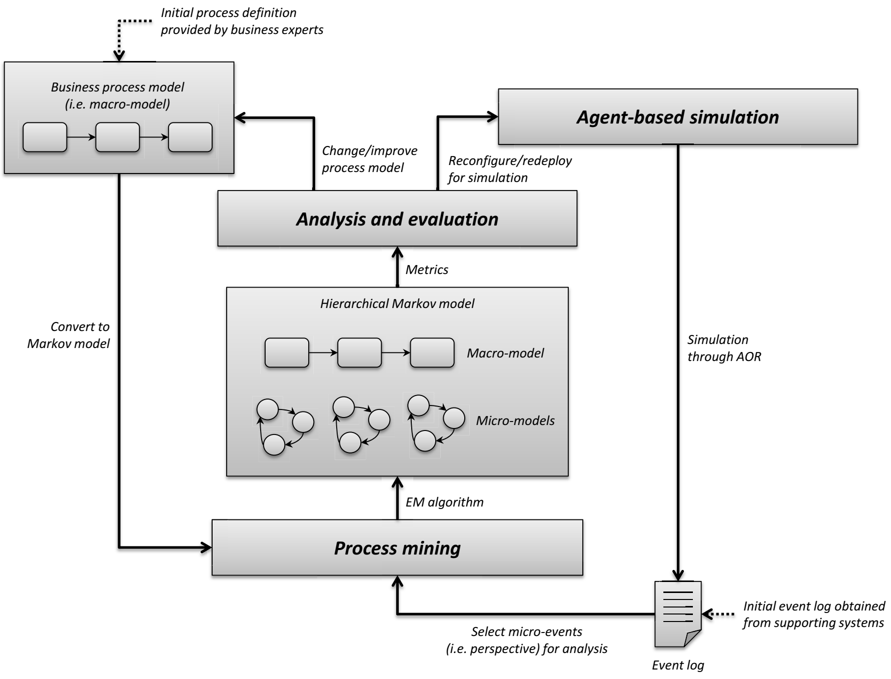

Initial process definition provided by business experts
Business process model (i.e. macro-model)
Agent-based simulation
Change/improve process model
Reconfigure/redeploy for simulation
Analysis and evaluation
Metrics
Hierarchical Markov model
Macro-model
Micro-models
Convert to Markov model
Simulation through AOR
EM algorithm
Process mining
Initial event log obtained from supporting systems
Select micro-events (i.e. perspective) for analysis
Event log

Fig. 1. Business process improvement cycle

From such hierarchical model, it is possible to assess whether the high-level process model is actually a good representation for the observed low-level behavior. In particular, it is possible to measure the quality of such model through a set of metrics. The use of metrics to evaluate the quality of process models has been thoroughly investigated in the literature [11–14], but here we focus on a set of metrics that are more tailored to the hierarchical model that is at the core of our approach. Specifically, such hierarchical model should be "balanced" in the sense that all of its components – i.e. the micro-models and the macro-model – should have a similar complexity, rather than having some components that are extremely complex and others that are oversimplified.

Following the analysis and evaluation of the hierarchical model through a set of metrics, the analyst can change the high-level description of the process in order to create a better representation for the behavior that actually occurs in practice. However, changing the high-level process may cause a different perception of how the low-level events map to the high-level activities. For example, if an activity is split in two, or if two activities are merged together, the relationship between the low-level events and the high-level activities will change. Contrary to what may appear at first sight, such change is not entirely predictable, since agents organize themselves in a non-deterministic way to carry out the process. In addition, process mining techniques do not provide perfect accuracy, so one can gain further insight into the run-time behavior of the process by trying out different configurations for the process model.

Ideally, a new version of the process would be deployed in the organization and one would be able to collect new event logs. Then one could run the process mining algorithm again in order to check which changes have been introduced in the run-time behavior, and whether the new process is a good representation of that behavior. In practice, one may not be able to do such experiments on the real-world environment due to the risks, costs or time involved. Therefore, we introduce the possibility of carrying out an agent-based simulation for the new process model. This simulation will be configured with the knowledge that has been collected so far about the process.

In previous work [10], we have shown that it is possible to generate event logs from agent-based simulations. In particular, we used the AOR framework [9] for that purpose. With this platform, it is possible to configure simulation scenarios which comprise multiple agents interacting with each other. An event log of these interactions can be recorded and used for process mining analysis. Together with the high-level description of the business process, this event log can be used to extract a new hierarchical model and again analyze and evaluate this model in order to assess the opportunity for further changes. This improvement cycle is based on simulation and mining, and it can be repeated until a satisfactory process model is found.

3 Agent-Based Simulation

Agent-based simulation (ABS) [6–8] focuses on the analysis of business systems involving interactions among agents. ABS can be used to represent organizational settings in a natural way, since they involve business actors that communicate and interact with each other. In particular, it is possible to use ABS for simulating the execution of a business process, thereby generating an event log. In our approach, we use the AOR simulation framework [9], which provides both high-level constructs (such as activities) and low-level constructs (such as incoming and outgoing message events) to facilitate the mapping of a business process model into a simulation model.

In the AOR framework, agents react to events in their environment by performing actions and by interacting with each other. There are basically two different kinds of events:

- An *exogenous event* is an external event (such as the arrival of a new customer) which is not caused by any previous event. Usually, the occurrence of such an event triggers a new instance of the business process. To run multiple instances of a business process, the AOR system schedules several exogenous events to trigger the process at different points in time.
- The second kind of event is a *caused event*. For example, if agent X sends a message M1 to agent Y, then this may result in another message M2 being sent from agent Y to agent Z. Agents send messages to one another for different purposes, e.g. for reporting, for making requests and for responding. The chaining of messages through caused events is what keeps the simulation running until there are no more events.

The specification of a simulation scenario begins by defining a set of *entity types*, including different types of agents, messages and events. The behavior of agents is specified by means of *reaction rules*. Typically, such a rule defines that when a certain message is received, the information state of the agent is updated and another message is sent to some other agent. Since the rules for each agent are defined separately, the simulation scenario is effectively implemented in a decentralized way by the combined behavior of all agents.

A second kind of rule in an AOR simulation scenario are *environment rules*. While reaction rules define the behavior of agents, environment rules define the behavior associated with the external environment. An environment rule specifies that when an exogenous event occurs, the state of certain objects is changed and certain follow-up events result from it. For example, an environment rule may specify that when a certain event occurs, a message is sent to an agent; sending this message then triggers a reaction rule of the receiving agent, creating a chain of events that puts the simulation in motion.

Environment rules also have the ability to create (or destroy) agents. This is especially useful to simulate, for example, the arrival (or departure) of customers. A set of *initial conditions* for the simulation scenario specifies which agents already exist in the scenario at the beginning of the simulation. The initial conditions also include a schedule for the occurrence of at least one exogenous event to trigger the simulation.

All of these constructs (i.e. entity types, reaction rules, environment rules, and initial conditions) are specified using an XML-based language called *AOR Simulation Language* (AORSL) [15]. The specification of a simulation scenario in AORSL is transformed into Java code by the AOR system. Running the simulation amounts to running this auto-generated Java code.

4 Process Mining with Hierarchical Markov Models

Process mining [5] includes a number of different perspectives, namely the control-flow perspective, the organizational perspective, and the performance perspective. In each of these perspectives there are a number of specialized techniques, such as the α-algorithm [16] for the control-flow perspective, the social network miner [17] for the organizational perspective, or the dotted chart [18] for the performance perspective. However, these techniques have in common the fact that they work at the same level of abstraction as the events recorded in the event log, so they may not be very helpful in providing insight in terms of the high-level activities that are used to describe the business processes.

More recently, there has been some effort in developing techniques that are able to address this problem. Some of these techniques work by extracting a low-level model from the event log and then creating a more abstract representation of that model [19, 20]. Other techniques begin by translating the event log into a more abstract sequence of events, and then extracting models from that translated event log [21, 22]. Here we address the problem with a special kind of model – i.e. the hierarchical Markov model introduced in [10] – which is able to capture both the high level of abstraction at which the process is defined and the low-level behavior that can be observed in the event log.

Figure 2 illustrates a simple example of a hierarchical Markov model. Here, the business process is described on a high level as comprising the sequence of activities A, B, and C. When each of these activities is performed, this results in some sequence of low-level events being recorded in the event log. Figure 2 represents these low-level events as X, Y and Z. In practice, these may represent the actions that are being carried out by agents, or they may also represent the interactions, i.e. the messages exchanged between agents. The meaning of X, Y and Z depends on the type of events that are recorded in the event log.

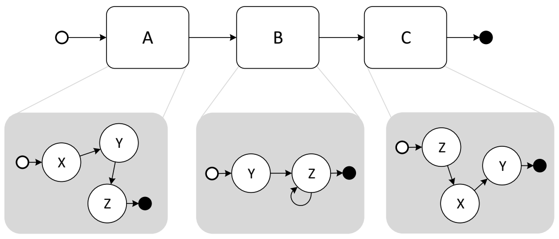


Fig. 2. A simple hierarchical Markov model

The hierarchical model in Figure 2 means that executing activity A results in the sequence of events XYZ being recorded in the event log. In a similar way, activity B results in a sequence of events in the form YZZ..., where there may be multiple Z's until a certain condition becomes true. Finally, activity C results in a sequence of events in the form ZXY. Both the high-level process and these sequences of low-level events are represented as Markov chains. Executing this model corresponds to performing the sequence of activities ABC. However, in the event log we find sequences of events such as XYZYZZZXY.

The sequence ABC is called the *macro-sequence*, and the high-level Markov chain that represents the business process expressed in terms of the activities A, B, and C is referred to as the *macro-model*. On the other hand, the sequence of events XYZYZZZXY is called a *micro-sequence*, and the low-level Markov chains that describe the behavior of each macro-activity in terms of the events X, Y and Z are referred to as a the *micro-models*.

The hierarchical Markov model relates to the approach depicted in Figure 1 in the following way: the high-level description of the business process corresponds to the macro-model, and the low-level event log corresponds to the micro-sequence.[1] Therefore, both the macro-model and the micro-sequence are known. The problem is how to discover the macro-sequence and the micro-models from the given macro-model and micro-sequence.

This problem has been addressed in detail in [10], and it can be solved with an Expectation-Maximization (EM) procedure [23] as follows:

1. Draw one macro-sequence at random from the macro-model (in the example of Figure 2 only one macro-sequence is possible: ABC).
2. From the macro-sequence and the micro-sequence, find an estimate for the micro-models (Algorithm 2 in [10]).
3. From the macro-model and the micro-models from step 2, find the most likely macro-sequence (Algorithm 3 in [10]).
4. With the macro-sequence from step 3, repeat step 2. Then with the micro-models from step 2, repeat step 3. Do this until the micro-models converge.

In [24] the authors show that this EM procedure is able to deal with workflow patterns such as branching, parallelism and loops. Our goal here is just to highlight that from a high-level model of the business process (i.e. the macro-model) and a low-level event log recorded during execution (i.e. the micro-sequence), it is possible to derive a set of micro-models that capture the low-level behavior within each high-level activity, as depicted in Figure 2. The initial macro-model, together with the recently discovered micro-models, can then be evaluated through a set of metrics, as explained in the next section.

5 Evaluation Metrics

In the literature, there are several metrics that have been proposed for the analysis of business process models [11–14]. In some cases, these metrics have their origin in network analysis and software engineering.

Metrics for describing network structure are usually derived from graph theory and network theory. For example, there are some characteristics that are normally used to describe a network structure, e.g. centrality, degree, density, and connectivity [25]. A business process model, expressed in BPMN for example, can be viewed as a special kind of graph, and therefore those network analysis techniques can be applied in this context as well.

In software engineering, there are also several metrics to measure aspects that are relevant for quality assurance. These metrics depend on the programming

[1] More precisely, an event log usually contains multiple traces, and each trace corresponds to a separate micro-sequence.

paradigm being used. In procedural programming, the most common metrics are structural and they are based on the counting of functions along the control-flow, such as the cyclomatic number [26], the information flow metric (fan-in, fan-out) [27], and the COnstructive COst MOdel [28], among others. In object-oriented programming, there is a different set of metrics to measure factors such as cohesion and coupling [29]. To some extent, some of these metrics can be used to evaluate business process models too.

In this work, we are interested in metrics that can be used to evaluate process models expressed as hierarchical Markov models. For this purpose, we studied a number of metrics focusing on factors such as size, density, modularity, control-flow, etc. [11, 30, 31]. The literature provides a lot of metrics to measure these factors, but here we selected a subset of metrics that are more geared towards the structure and complexity of those models. In particular, we are interested in metrics that allow us to determine if the hierarchical model is "balanced" in the sense that the macro-model and the micro-models should have similar complexity. The chosen metrics are summarized in Table 1.

Table 1. Metrics to evaluate the components of a hierarchical model

Metric	**Abbrev.**	**Focus**	**Definition**
No. of arcs per node [11]	NAN	Density	Ratio of number of arcs to number of nodes.
Relational density [32]	RD	Density	Ratio of number of arcs to the total number of possible arcs.
No. of paths [26]	NP	Control-flow	Number of all possible distinct paths between start node and end node.
Path length [25]	PL	Size	Number of nodes between start node and end node. (For a given model, this is calculated as the average length of all possible paths.)
Cyclomatic complexity [26]	CC	Control-flow	Number of linearly-independent paths. Can be measured as $M = E - N + 2P$ where E is the number of arcs, N is the number of nodes, and P is the number of exit nodes.
Fan-in/Fan-out [14]	FIO	Modularity	$(f_{in} \cdot f_{out})^2$ where f_{in} is the number of nodes that precede a given node, and f_{out} is the number of nodes that follow a given node. (For a given model, f_{in} and f_{out} are calculated separately as an average across all nodes.)
No. of sub-processes	SUB	Modularity	Number of nested sub-processes in a model.

All metrics (except SUB) can be calculated separately for the macro-model and for each micro-model. This provides an assessment of each component in the hierarchical model. For example, in the hierarchical model of Figure 2 there are four components, and the metrics can be calculated for each of them. To get an assessment of the hierarchical model as a whole, in this work we take a simple average of the results obtained across all components.

6 Application Scenario

The purpose of this application scenario is to illustrate how the proposed approach can be used to facilitate the refinement of a process model, so that this model becomes not only a more precise representation of the observed behavior, but also a model which achieves better results in terms of the metrics described in the previous section. The scenario involves a purchase process which is described on a high-level both in text and by means of a BPMN diagram. Due to space restrictions, we will show the initial version (version 1) and the final version (version 2), but there could be other versions in between.

Initially, the process is described as follows:

> In a company, an employee needs a certain commodity (e.g. a printer cartridge) and submits a request for that product to the warehouse. If the product is available at the warehouse, then the warehouse dispatches the product to the employee. Otherwise, the purchasing department buys the product, and then the warehouse dispatches the product to the employee.

This process is represented as a BPMN diagram in Figure 3, and its control flow can be expressed as a Markov chain with transition probabilities, as shown in Figure 4. In this macro-model, there are just three high-level activities and one decision. For lack of further info, we assume that the two possible outcomes from this decision are equally likely, so the transition probabilities from the "Requisition" activity to "Dispatch Product" and "Buy Product" are both 0.5 in Figure 4. In this Markov chain, there are also two special states (○ and ●) to represent the beginning and end of the process, respectively.

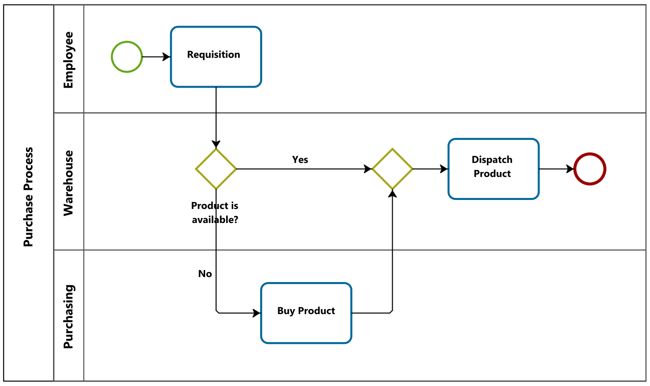


Fig. 3. BPMN diagram for the purchase process (version 1)

In a real scenario, an initial event log can be obtained from the supporting systems in order to capture the low-level operations performed by the participants in the process, or the interactions (i.e. message exchanges) between those participants. Table 2 shows an example of the later. Here, each message has

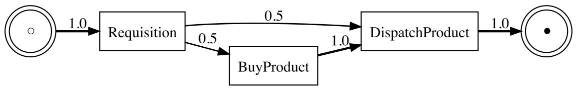


Fig. 4. Markov chain representation for the purchase process (version 1)

been represented as having a certain type or meaning. For example, StockRequest is a message sent from the Employee to the Warehouse. While this message appears to be related to the "Requisition" activity, for other messages the relationship to a high-level activity may not be so clear.

Table 2. Excerpt of an event log

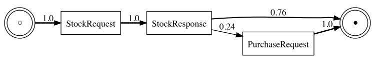
case id	*sender*	*message*	*receiver*	*timestamp*
1	Employee	StockRequest	Warehouse	2013-02-02 11:26
1	Warehouse	StockResponse	Employee	2013-02-04 16:07
1	Employee	FetchProduct	Warehouse	2013-02-05 08:54
1	Warehouse	ProductReady	Employee	2013-02-06 10:23
1	Employee	ProductReceived	Warehouse	2013-02-07 15:47
2	Employee	StockRequest	Warehouse	2013-02-12 09:31
2	Warehouse	StockResponse	Employee	2013-02-14 14:10
2	Employee	PurchaseRequest	Purchasing	2013-02-15 16:35
2	Purchasing	InfoRequest	Employee	2013-02-18 17:21
2	Employee	InfoResponse	Purchasing	2013-02-19 10:52
2	Purchasing	ApprovalResult	Employee	2013-02-20 12:05
2	Purchasing	PurchaseOrder	Supplier	2013-02-21 15:19
...	...	...	...	...

From the event log in Table 2 it is possible to retrieve the sequence of events (i.e. micro-sequence) for each process instance (i.e. case id). Also it is possible to select one of the columns *sender*, *message* and *receiver* for analysis. Here we will use the *message* column, so the micro-sequence for case 1 is:

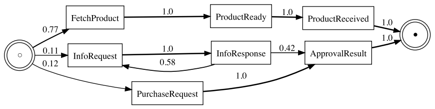
StockRequest→StockResponse→FetchProduct→ProductReady→ProductReceived

The micro-sequences for other cases can be extracted in a similar way. These micro-sequences, together with the macro-model in Figure 4, are provided as input to the algorithm described in Section 4, which produces the micro-models shown in Figure 5. Note that in Figure 5 there is some duplicated behavior in the "Dispatch product" and "Buy product" activities, resulting in longer and more complex micro-models. These micro-models were evaluated using the metrics defined in Section 5 and the results are shown in Table 3.

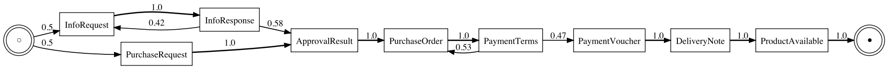
The results in Table 3 reflect the fact that the micro-models in Figure 5 are relatively more complex when compared to the macro-model in Figure 4. In particular, the metrics NP (no. of paths) and PL (path length) are significantly higher for the micro-models when compared to the same metrics for the macro-model. This means that the hierarchical model is somewhat "unbalanced", in the sense that the macro-model is of significant less complexity than the micro-models, and therefore it is probably an over-simplified representation of the



(a) Requisition



(b) Dispatch product



(c) Buy product

Fig. 5. Micro-models for the purchase process (version 1)

Table 3. Metrics applied to the purchase process (version 1)

	Metrics						
Model	NAN	RD	NP	PL	CC	FIO	SUB
Macro model	1.00	0.33	2.00	2.50	2.00	3.13	–
Micro Model (Requisition)	1.00	0.33	2.00	2.50	2.00	1.77	–
Micro Model (Dispatch Product)	1.22	0.17	4.00	3.25	4.00	2.16	–
Micro Model (Buy Product)	1.18	0.13	6.00	9.33	4.00	2.63	–
Complete model (avg.)	1.10	0.24	3.25	4.40	3.00	2.42	3

business process. There appears to be much more behavior in the micro-models than what the macro-model is able to account for.

Therefore, the business experts are encouraged to describe the process in more detail. Such description could be as follows:

> In a company, an employee needs a certain commodity (e.g. a printer cartridge) and submits a request for that product to the warehouse. If the product is available at the warehouse, then the warehouse dispatches the product to the employee. Otherwise, the product must be purchased from an external supplier. All purchases must be approved by the purchasing department. If the purchase is not approved, the process ends at that point. On the other hand, if the purchase is approved, the purchasing department orders and pays for the product from the supplier. The supplier delivers the product to the warehouse, and the warehouse dispatches the product to the employee.

This new version of the process is depicted in Figure 6 and is expressed as a Markov chain (i.e. macro-model) in Figure 7. On the other hand, to mine the hierarchical model, we will need an event log with the run-time behavior for this process. Rather than deploying the model in the organization and waiting for an event log to be recorded over a long period of time, we use the AOR framework to build an agent-based simulation scenario based on the previous version of the process (i.e. the hierarchical model in Figures 4 and 5).

If the behavior of the high-level process had been changed in this new version (e.g. if two high-level activities had been switched in their execution order) then we would need to adapt the simulation scenario in order to reflect those changes. However, here the new macro-model is just a refinement of the previous one, so

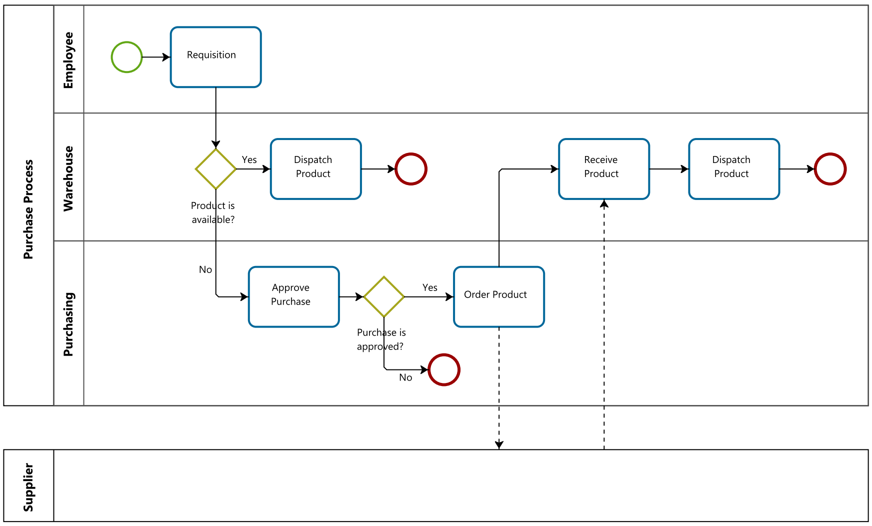


Fig. 6. High-level description of the purchase process (version 2)

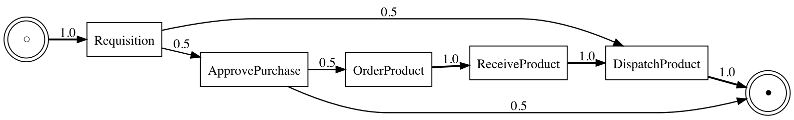
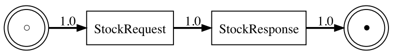


Fig. 7. Markov chain representation for the purchase process (version 2)

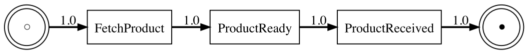
we can use the previous hierarchical model as a basis for simulation, without the need for further changes. Even though the hierarchical model in Figures 4 and 5 is not a balanced model, for simulation purposes it is still a perfectly valid model for reproducing the behavior of the process.

By running the simulation scenario in AOR, it is possible to collect a new and possibly larger event log to mine the next version of the hierarchical model. The event logs that are generated by the AOR framework are in XML, but they can be easily converted to the tabular form of Table 2. In this experiment, we ran a simulation with 10,000 steps, which produced an event log with 140 process instances and a total of 1136 events. The event log, together with the macro-model of Figure 7, were provided as input to the algorithm of Section 4, which produced the micro-models shown in Figure 8.

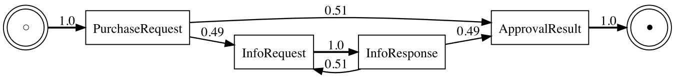
This new hierarchical model (i.e. Figures 7 and 8) was evaluated using the same metrics as before. As can be seen in Table 4, this new model is more balanced because the micro-models are, in general, of significantly lower complexity than before, while the macro-model is only slightly more complex. Comparing the last row of Table 4 with the last row of Table 3 shows that, overall, the new model is less complex than the previous one. In particular, `NP` (no. of paths),



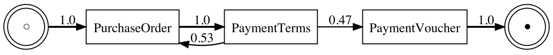
(a) Requisition



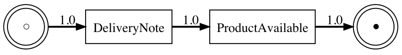
(b) Dispatch product



(c) Approve purchase



(d) Order product



(e) Receive product

Fig. 8. Micro-models for the purchase process (version 2)

Table 4. Metrics applied to the purchase process (version 2)

	Metrics						
Model	NAN	RD	NP	PL	CC	FIO	SUB
Macro model	1.14	0.23	3.00	3.00	3.00	2.82	–
Micro Model (Requisition)	0.75	0.38	1.00	2.00	1.00	1.00	–
Micro Model (Dispatch Product)	0.80	0.27	1.00	3.00	1.00	1.00	–
Micro Model (Approve Purchase)	1.17	0.29	3.00	4.00	3.00	5.06	–
Micro Model (Order Product)	1.00	0.33	2.00	4.00	2.00	3.13	–
Micro Model (Receive Product)	0.75	0.38	1.00	2.00	1.00	1.00	–
Complete model (avg.)	0.93	0.31	1.83	3.00	1.83	2.34	5

PL (average path length), and CC (cyclomatic complexity) are significantly lower than before. This means that the model in Figure 6 is not only a more accurate description of the process, but also the low-level behavior associated with each high-level activity is simpler and easier to understand.

7 Conclusion

In this work we described an iterative approach for the improvement of business process models based on process mining and agent-based simulation. The need for such approach is justified by the fact that there is a gap between the high-level of abstraction at which processes are usually modeled and the low-level nature of events that are generated during execution. The process mining technique that we used here is able to capture this relationship in the form of a hierarchical Markov model. On the other hand, an agent-based simulation platform is used as a means to generate the low-level behavior for new versions of the process. Provided with a set of metrics that serve as guidance, the process analyst can insert changes to the process model, reconfigure the simulation platform, generate a new event log, and mine a new hierarchical model that again captures the relationship between the high-level activities and the low-level behavior. Doing this iteratively will lead to a better model, where "better" means more accurate, more balanced, less complex, and easier to understand.

Due to space restrictions, here we focused on the analysis of the control-flow alone, but the same approach can be applied to the analysis of the organizational perspective, which includes the handover of work between agents and the collaboration of agents within each case. This can be done by selecting other columns for analysis, namely the *sender* column or the *receiver* column in the event log of Table 2. In future work, we are planning to improve several aspects of the proposed approach, namely establishing guidelines for the conversion of BPMN models to Markov models, supporting the automatic generation of an AOR simulation scenario from a given hierarchical model, and expanding the set of metrics used in the analysis and evaluation phase.

References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0 (2011)
2. Scheer, A.W.: ARIS: Business Process Modeling, 3rd edn. Springer (2000)
3. van der Aalst, W.M.P.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
4. Weske, M.: Business Process Management: Concepts, Languages, Architectures. 2nd edn. Springer (2012)
5. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
6. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. PNAS 99(suppl. 3), 7280–7287 (2002)
7. Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., Persson, M.: Applications of agent based simulation. In: Antunes, L., Takadama, K. (eds.) MABS 2006. LNCS (LNAI), vol. 4442, pp. 15–27. Springer, Heidelberg (2007)
8. Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: Review and development recommendations. Simulation 82(9), 609–623 (2006)
9. Wagner, G.: AOR modelling and simulation: Towards a general architecture for agent-based discrete event simulation. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS (LNAI), vol. 3030, pp. 174–188. Springer, Heidelberg (2004)
10. Ferreira, D.R., Szimanski, F., Ralha, C.G.: A hierarchical Markov model to understand the behaviour of agents in business processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 150–161. Springer, Heidelberg (2013)
11. Mendling, J.: Metrics for Process Models. LNBIP, vol. 6. Springer, Heidelberg (2009)
12. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Quality Metrics for Business Process Models. In: 2007 BPM & Workflow Handbook, pp. 179–190. Future Strategies Inc. (2007)
13. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems 36(2), 498–516 (2011)

14. Gruhn, V., Laue, R.: Approaches for Business Process Model Complexity Metrics. In: Technologies for Business Information Systems, pp. 13–24. Springer (2007)
15. Nicolae, O., Wagner, G., Werner, J.: Towards an executable semantics for activities using discrete event simulation. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009 Workshops. LNBIP, vol. 43, pp. 369–380. Springer, Heidelberg (2010)
16. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16, 1128–1142 (2004)
17. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for organizational mining. Decision Support Systems 46(1), 300–317 (2008)
18. Song, M., van der Aalst, W.M.P.: Supporting process mining by showing events at a glance. In: Proceedings of 17th Annual Workshop on Information Technologies and Systems, pp. 139–145 (2007)
19. Greco, G., Guzzo, A., Pontieri, L.: Mining hierarchies of models: From abstract views to concrete specifications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 32–47. Springer, Heidelberg (2005)
20. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
21. Günther, C.W., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009 Workshops. LNBIP, vol. 43, pp. 128–139. Springer, Heidelberg (2010)
22. Bose, R.P.J.C., Verbeek, E.H.M.W., van der Aalst, W.M.P.: Discovering hierarchical process models using ProM. In: Nurcan, S. (ed.) CAiSE Forum 2011. LNBIP, vol. 107, pp. 33–48. Springer, Heidelberg (2012)
23. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. Wiley Series in Probability and Statistics. Wiley-Interscience (2008)
24. Ferreira, D.R., Szimanski, F., Ralha, C.G.: Mining the low-level behavior of agents in high-level business processes. International Journal of Business Process Integration and Management (to appear, 2013)
25. Newman, M.: The structure and function of complex networks. SIAM Review 45(2), 167–256 (2003)
26. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering 2(4), 308–320 (1976)
27. Henry, S.M., Kafura, D.G.: Software structure metrics based on information flow. IEEE Transactions on Software Engineering 7(5), 510–518 (1981)
28. Boehm, B.W.: Software engineering economics. Prentice-Hall (1981)
29. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
30. Nissen, M.E.: Redesigning reengineering through measurement-driven inference. MIS Quarterly 22(4), 509–534 (1998)
31. Cardoso, J.: Control-flow complexity measurement of processes and Weyuker's properties. In: 6th International Enformatika Conference. Transactions on Enformatika, Systems Sciences and Engineering, vol. 8, pp. 213–218 (October 2005)
32. Vanderfeesten, I.T.P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.: On a quest for good process models: The cross-connectivity metric. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 480–494. Springer, Heidelberg (2008)