Predicting Process Behavior in WoMan

Conference Paper *in* Lecture Notes in Computer Science · November 2016
DOI: 10.1007/978-3-319-49130-1_23

4 authors:

Stefano Ferilli
Università degli Studi di Bari Aldo Moro
331 PUBLICATIONS **1,508** CITATIONS
SEE PROFILE

Esposito Floriana
Università degli Studi di Bari Aldo Moro
598 PUBLICATIONS **4,047** CITATIONS
SEE PROFILE

Domenico Redavid
Università degli Studi di Bari Aldo Moro
53 PUBLICATIONS **178** CITATIONS
SEE PROFILE

Sergio Angelastro
Università degli Studi di Bari Aldo Moro
8 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Predicting Process Behavior in WoMan

Stefano Ferilli[1,2(✉)], Floriana Esposito[1,2], Domenico Redavid[3], and Sergio Angelastro[1]

[1] Dipartimento di Informatica, Università di Bari, Bari, Italy
{stefano.ferilli,floriana.esposito,sergio.angelastro}@uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni, Università di Bari, Bari, Italy
[3] Artificial Brain S.r.l., Bari, Italy
redavid@abrain.it

Abstract. In addition to the classical exploitation as a means for checking process enactment conformance, process models may be precious for making various kinds of predictions about the process enactment itself (e.g., which activities will be carried out next, or which of a set of candidate processes is actually being executed). These predictions may be much more important, but much more hard to be obtained as well, in less common applications of process mining, such as those related to Ambient Intelligence. Also, the prediction performance may provide indirect indications on the correctness and reliability of a process model. This paper proposes a way to make these kinds of predictions using the WoMan framework for workflow management, that has proved to be able to handle complex processes. Experimental results on different domains suggest that the prediction ability of WoMan is noteworthy and may be useful to support the users in carrying out their processes.

Keywords: Process mining · Activity prediction · Process prediction

1 Introduction

A classical application domain of process management is the industrial one, where the activities of a production process must be monitored and checked for compliance with a desired behavior. If a formal model of the desired behavior is available, the supervision task may be automated, provided that the events related to the process enactment can be detected and delivered to the automatic system. Given an intermediate status of a process execution, knowing how the execution will proceed might allow the (human or automatic) supervisor to take suitable actions that facilitate the next activities. However, it may be expected that in an industrial environment the rules that determine how the process must be carried out are quite strict. So, the emphasis is more on conformance checking, while prediction of process evolution is more trivial. This situation changes significantly if we move toward other, less traditional application domains for process management, that have been introduced more recently. For instance, considering the daily routines of people, at home or at work, as a process, much

G. Adorni et al. (Eds.): AI*IA 2016, LNAI 10037, pp. 308–320, 2016.
DOI: 10.1007/978-3-319-49130-1_23

more variability is involved, and obtaining reliable predictions becomes both more complicated and more useful.

Another shortcoming in this landscape is that, due to the complexity of some domains, manually building the process models that are to be used for supervision, compliance checking, and prediction is very complex, costly, and error-prone. The solution of automatically learning these models from examples of actual execution, which is the task of Process Mining [1,2], opens the question of how to validate the learned models to ensure that they are correct and effective. Indeed, models are learned automatically exactly because the correct desired model is not available, and thus such a validation can be carried out only empirically by applying the learned model to future process enactments. In addition to check *a posteriori* the correctness of new process enactments based on the learned model, an investigation of how good it is in providing hints about what is going on in the process execution, and what will happen next, is another, very relevant and interesting, way to assess the quality of the model.

This paper proposes two approaches for process-related predictions. The former tackles the more classical setting in which the process (and the corresponding model) that is being enacted is known, and one wishes to predict which activities will be carried out next at any moment during the process execution. The latter is more original. It assumes that one is supervising the enactment of an unknown process, and that a set of candidate processes (and corresponding models) is available, among which one aims at predicting which one is being actually enacted. The proposed solutions work in the WoMan framework for process management [3], that introduced significant advances to the state-of-the-art. Specifically, it adopts a representation formalism that, while being more powerful than Petri/Workflow Nets, makes the above predictions more complicated. This paper investigates the reasons for such an increased complexity, and leverages their peculiarities for obtaining useful hints for making the predictions.

It is organized as follows. The next section recalls some basics on process management and the WoMan framework. Then, the proposed approaches to making predictions are presented in Sect. 4 and evaluated in Sect. 5. Finally, in the last section, we draw some conclusions and outline future work issues.

2 Process Management Basics and the WoMan Framework

A *process* consists of actions performed by agents (humans or artifacts) [4,5]. A *workflow* is a formal specification of how these actions can be composed to result in valid processes. Allowed compositional schemes include sequential, parallel, conditional, or iterative execution [6]. A process execution can be described in terms of *events*, i.e. identifiable, instantaneous actions (including decisions upon the next activity to be performed). A *case* is a particular execution of actions compliant to a given workflow. Case *traces* consist of lists of events associated to *steps* (time points) [7]. A *task* is a generic piece of work, defined to be executed for many cases of the same type. An *activity* is the actual execution of a task

by a *resource* (an agent that can carry it out). Relevant events are the start and end of process executions, or of activities [5].

The WoMan framework [3,8] lies at the intersection between *Declarative* Process Mining [9] and Inductive Logic Programming (ILP) [10]. It introduced some important novelties in the process mining and management landscape. A fundamental one is the pervasive use of First-Order Logic as a representation formalism, that provides a great expressiveness potential and allows one to describe contextual information using relationships. Experiments proved that it is able to handle efficiently and effectively very complex processes, thanks to its powerful representation formalism and process handling operators. In the following, we briefly and intuitively recall its fundamental notions.

WoMan's learning module, **WIND** (Workflow INDucer), allows one to learn or refine a process model according to a case, after the case events are completely acquired. The refinement may affect the structure and/or the probabilities. Differently from all previous approaches in the literature, it is *fully incremental*: not only can it refine an existing model according to new cases whenever they become available, it can even start learning from an empty model and a single case, while others need a (large) number of cases to draw significant statistics before learning starts. This is a significant advance with respect to the state-of-the-art, because continuous adaptation of the learned model to the actual practice can be carried out efficiently, effectively and transparently to the users [8].

According to [4,11], WoMan takes as input trace elements consisting of 7-tuples $\langle T, E, W, P, A, O, R \rangle$, where T is the event timestamp, E is the type of the event (one of 'begin_process', 'end_process', 'begin_activity', 'end_activity'), W is the name of the reference workflow, P is the case identifier, A is the name of the activity, O is the progressive number of occurrence of that activity in that case, and R (optional) specifies the resource that is carrying out the activity.

A model describes the structure of a workflow using the following elements:

tasks: the kinds of activities that are allowed in the process;
transitions: the allowed connections between activities (also specifying the involved resources).

The core of the model, carrying the information about the flow of activities during process execution, is the set of transitions. A transition $t : I \Rightarrow O$, where I and O are multisets of tasks, is enabled if all input tasks in I are active; it occurs when, after stopping (in any order) the concurrent execution of all tasks in I, the concurrent execution of all output tasks in O is started (again, in any order). For analogy with the notions of 'token' and 'marking' in Petri Nets, during a process enactment we call a *token* an activity that has terminated and can be used to fire a transition, and a *marking* the set of current tokens. Both tasks and transitions are associated to the multiset C of training cases in which they occurred (a multiset because a task or transition may occur several times in the same case, if loops or duplicate tasks are present in the model). It can be exploited both during the conformance check of new cases (to ensure that the flow of activities in the new case was encountered in at least one training case) and for computing statistics on the use of tasks and transitions. In particular,

it allows us to compute the probability of occurrence of a task/transition in a model learned from n training cases as the relative frequency $|C|/n$.

As shown in [3,8], this representation formalism is more powerful than Petri or Workflow Nets [1], that are the current standard in Process Mining. It can smoothly express complex task combinations and models involving invisible or duplicate tasks, which are problematic for those formalisms. Indeed, different transitions can combine a given task in different ways with other tasks, or ignore a task when it is not mandatory for a specific passage. Other approaches, by imposing a single component for each task, route on this component all different paths passing from that task, introducing combinations that were never seen in the examples.

The increased power of WoMan's representation formalism for workflow models raises some issues that must be tackled. In Petri Nets, since a single graph is used to represent the allowed flow(s) of activities in a process, at any moment in time during a process enactment, the supervisor knows which tokens are available in which places, and thus he may know exactly which tasks are enabled. So, the prediction of the next activities that may be carried out is quite obvious, and checking the compliance of a new activity with the model means just checking that the associated task is in the set of enabled ones. Conversely, in WoMan the activity flow model is split into several 'transitions', and different transitions may share some input and output activities, which allows them to be composed in different ways with each other. As a consequence, many transitions may be eligible for application at any moment, and when a new activity takes place there may be some ambiguity about which one is actually being fired. Such an ambiguity can be resolved only at a later time. Let us see this through an example. Suppose that our model includes, among others, the following transitions:

$$t_1 : \{x\} \Rightarrow \{a, b\} \quad ; \quad t_2 : \{x, y\} \Rightarrow \{a\} \quad ; \quad t_3 : \{w\} \Rightarrow \{d, a\}$$

and that the current marking (i.e., the set of the latest activities that were terminated in the current process enactment, without starting any activity after their termination) is $\{x, y, z\}$. Now, suppose that activity a is started. It might indicate that either transition t_1 or transition t_2 have been fired. Also, if an activity d is currently being executed due to transition t_3, the current execution of a might correspond to the other output activity of transition t_3, which we are still waiting to happen to complete that transition. We call each of these alternatives a *status*. This ambiguity about different statuses that are compliant with a model at a given time of process enactment must be properly handled when supervising the process enactment, as we will show later.

3 Workflow Supervision

The supervision module allows one to check whether new cases are compliant with a given model. **WEST** (Workflow Enactment Supervisor and Trainer) takes the case events as long as they are available, and returns information about their compliance with the currently available model for the process they refer to. The

output for each event can be 'ok', 'error' (e.g., when closing activities that had never begun, or terminating the process while activities are still running), or a set of warnings denoting different kinds of deviations from the model (e.g., unexpected task or transition, preconditions not fulfilled, unexpected resource running a given activity, etc.).

Now, as we have pointed out in the previous section, given an intermediate status of the process enactment and a new activity that is started, there may be different combinations of transitions that are compliant with the new activity, and one may not know which is the correct one until a later time. Thus, all the corresponding alternate evolutions of the status must be carried on by the system. Considering again the previous example, each of the three proposed options would change in a different way the status of the process, as follows:

t_1**:** firing this transition would consume x, yielding the new marking $\{y, z\}$ and causing the system to wait for a later activation of b;
t_2**:** firing this transition would consume x and y, yielding the new marking $\{z\}$ and causing the completion of transition t_2;
t_3**:** firing this transition would not consume any element in the marking, but would cause the completion of transition t_3.

When the next event is considered, each of these evolutions is a possible status of the process enactment. On one hand, it poses again the same ambiguity issues; on the other hand, it may point out that some current alternate statuses were wrong. So, as long as the process enactment proceeds, the set of alternate statuses that are compliant with the activities carried out so far can be both expanded with new branches, and pruned of all alternatives that become incompatible with the activities carried out so far.

Note also that each alternative may be compliant with a different set of training cases, and may rise different warnings (in the previous example, one option might be fully compliant with the model, another might rise a warning for task preconditions not fulfilled, and the other might rise a warning for an unexpected agent running that activity). WEST takes note of the warnings for each alternative and carries them on, because they might reflect secondary deviations from the model that one is willing to accept. Wrong alternatives will be removed when they will be found out to be inconsistent with later events in the process enactment. So, the question arises about how to represent each alternative status. As suggested by the previous example, we may see each status as a 5-tuple

$$\langle M, R, C, T, W \rangle$$

recording the following information:

M the marking, i.e., the set terminated activities that have not been used yet to fire a transition, each associated with the agent that carried it out and to the transition in which it was involved as an output activity;
R (for 'Ready') the set of activities that are ready to start, i.e., the output activities of transitions that have been fired in the status, and that the system is waiting for in order to complete those transitions;

C the set of training cases that are compliant with that status;
T the set of (hypothesized) transitions that have been fired to reach that status;
W the set of warnings raised by the various events that led to that status.

The system also needs to remember, at any moment in time, the set *Running* of currently running activities and the list *Transitions* of transitions actually carried out so far in the case. The set of statuses is maintained by WEST, as long as the events in a case are processed, according to Algorithm 1.

Algorithm 1. Maintenance of the structure recording valid statuses in WEST

Require: $\mathcal{M}$: process model
Require: *Statuses* : set of currently compliant statuses compatible with the case
Require: *Running* : set of currently running activities
Require: *Transitions*: list of transitions actually carried out so far
Require: $\langle T, E, W, P, A, O, R\rangle$: log entry
  if $E =$ begin_activity **then**
    $Running \leftarrow Running \cup \{A\}$
    for all $S = \langle M, R, C, T, P\rangle \in Statuses$ **do**
      $Statuses \leftarrow Statuses \setminus \{S\}$
      if $A \in R$ **then**
        $Statuses \leftarrow Statuses \cup \{\langle M, R \setminus \{A\}, C, T, P\rangle\}$
      end if
      for all $p : I \Rightarrow O \in \mathcal{M}$ **do**
        if $I \subseteq Marking \wedge A \in O$ **then**
          $C' \leftarrow C \cap C_p$ /* C_p training cases involving p */
          $P' \leftarrow P \cup P_{A,p,S}$ /* $P_{A,p,S}$ warnings raised by running A in p given S */
          $Statuses \leftarrow Statuses \cup \{\langle M \setminus I, R \cup (O \setminus \{A\}), C', T\&\langle t\rangle, P'\rangle\}$
        end if
      end for
    end for
  end if
  if $E =$ end_activity **then**
    if $A \notin Running$ **then**
      Error
    else
      $Running \leftarrow Running \setminus \{A\}$
      for all $S = \langle M, R, C, T, P\rangle \in Statuses$ **do**
        $S \leftarrow \langle M \cup \{A\}, R, C, T, P\rangle$
      end for
      if a transition t has been fully carried out **then**
        Transitions $\leftarrow$ Transitions $\&\langle t\rangle$
        for all $S = \langle M, R, C, T, P\rangle \in$ Statuses **do**
          if $T \neq$ Transitions **then**
            Statuses $\leftarrow$ Statuses $\setminus\{S\}$
          end if
        end for
      end if
    end if
  end if

4 Prediction Strategy

While in supervision mode, the prediction modules allow one to foresee which activities the user is likely to perform next, or to understand which process is being carried out among a given set of candidates. We recall that, due to the discussed set of alternate statuses that are compliant with the activities carried out at any moment, differently from Petri Nets it is not obvious to determine which are the next activities that will be carried out. Indeed, any status might be associated to different expected evolutions. The good news is that, having several alternate statuses, we can compute statistics on the expected activities in the different statuses, and use these statistics to determine a ranking of those that most likely will be carried out next.

Specifically, **SNAP** (Suggester of Next Action in Process) hypothesizes which are the possible/appropriate next activities that can be expected given the current intermediate status of a process execution, ranked by confidence. Confidence here is not to be interpreted in the mathematical sense. It is determined based on a heuristic combination of several parameters associated with the possible alternate process statuses that are compliant with the current partial process execution. Specifically, the activities that can be carried out next in a given status are those included in the *Ready* component of that status, or those belonging to the output set of transitions that are enabled by the *Marking* component of that status. The status parameters used for the predictions are the following:

1. frequency of activities across the various statuses (activities that appear in more statuses are more likely to be carried out next);
2. number of cases with which each status is compliant (activities expected in the statuses supported by more training cases are more likely to be carried out next);
3. number of warnings raised by the status (activities expected in statuses that raised less warnings are more likely to be carried out next);
4. confidence of the tasks and transitions as computed by the multiset of cases supporting them in the model (activities supported by more confidence, or belonging to transitions that are associated to more confidence, are more likely to be carried out next).

Finally, given a case of an unknown workflow, **WoGue** (Workflow Guesser) returns a ranking (by confidence) of a set of candidate process models. Again, this prediction is based on the possible alternate statuses identified by WEST when applying the events of the current process enactment to the candidate models. In this case, for each model, the candidate models are ranked by decreasing performance of their 'best' status, i.e. the status reporting best performance in (one or a combination of) the above parameters.

5 Evaluation

In the following, we evaluate the performance of the proposed prediction approaches on several datasets, concerning different kinds of processes. Two

are related to Ambient Intelligence, and specifically to the Ambient Assisted Living domain. The other concerns chess playing. In these domains, the prediction of the next activity that will be carried out is more complex than in industrial processes, because there is much more variability and subjectivity in the users' behavior, and there is no 'correct' underlying model, just some kind of 'typicality' can be expected. The chess domain also allows us to evaluate the performance of the process prediction approach.

5.1 Datasets

Our experiments were run on the following datasets:

Aruba was taken from the CASAS benchmark repository[1]. This dataset concerns people's daily routines, and includes continuous recordings of home activities of an elderly person, visited from time to time by her children, in a time span of 220 days. We considered each day as a case of the process representing the daily routine of the elderly person. Transitions, here, correspond to terminating some activities and starting new activities. The resources (i.e., which person performs which activity) are unknown.

GPItaly was built by extracting the data from one of the Italian use cases of the GiraffPlus project[2] [12]. It concerns, again, an elderly person in her home, but focusing on the movements of the home inhabitant(s) in the various rooms of the home. The dataset considered 253 days, each of which was, again, a case of the process representing the typical movements of people in the home. Tasks, here, correspond to rooms, while transitions correspond to leaving a room and entering another. The resource (i.e., the person that moves between the rooms) is always the same.

Chess consists of 400 reports of actual top-level matches played by Anatolij Karpov and Garry Kasparov (200 matches each) downloaded from the Italian Chess Federation website[3]. In this case, playing a chess match corresponds to enacting a process, where a task is the occupation of a specific square of the chessboard by a specific kind of piece (e.g., "black rook in a8" denotes the task of occupying the top-leftmost square with a black rook), and the corresponding activities are characterized by the time at which a piece starts occupying a square and the time at which the piece leaves that square. Matches are initialized by starting the activities corresponding to the initial positions of all pieces on the chessboard. Transitions correspond to moves: indeed, each move of a player terminates some activities (since it moves pieces away from the squares they currently occupy) and starts new activities (that is, the occupation by pieces of their destination squares)[4]. Given this perspective, the involved resources are the two players: 'white'

[1] http://ailab.wsu.edu/casas/datasets.html.
[2] http://www.giraffplus.eu.
[3] http://scacchi.qnet.it.
[4] Usually, each transition terminates one activity and starts another one. Special cases, such as captures and castling, may involve more pieces.

Table 1. Dataset statistics

	Cases	Events		Activities		Tasks		Transitions	
		Overall	Avg	Overall	Avg	Overall	Avg	Overall	Avg
Aruba	220	13788	62.67	6674	30.34	10	0.05	92	0.42
GPItaly	253	185844	369.47	92669	366.28	8	0.03	79	0.31
White	158	36768	232.71	18226	115.35	681	4.31	4083	25.84
Black	87	21142	243.01	10484	120.51	663	7.62	3006	34.55
Draw	155	32422	209.17	16056	103.59	658	4.25	3434	22.15

and 'black'. In this dataset we distinguished three processes, corresponding to the possible match outcomes: *white* wins, *black* wins, or *draw*.

Table 1 reports some statistics on the experimental datasets: number of cases, events, activities, tasks and transitions. The average number of events, activities, tasks, and transitions per case is also reported. It is apparent that each dataset is characterized by a peculiar mix of features. There are more cases for the Ambient Intelligence datasets than for the chess ones. However, the chess datasets involve many more different tasks and transitions, many of which are rare or even unique. In facts, the number of tasks and transitions is much less than the number of cases in the Ambient Intelligence datasets, while the opposite holds for the chess datasets. As regards the number of events and activities, GPItaly is the most complex one, followed by the chess datasets, while Aruba is the less complex one. The datasets are different also from a qualitative viewpoint. In Aruba, the cases feature many short loops (involving 1 or 2 activities), optional and duplicated activities. Concurrency is usually limited to at most two activities. In GPItaly, again, many short loops, optional and duplicated activities are present, but no concurrency. Finally, the chess datasets are characterized by very high parallelism: each game starts with 32 concurrent activities (a number which is beyond the reach of many current process mining systems [3]). This number progressively decreases as long as the game proceeds and pieces are taken, but remains still high (about 10 concurrent activities) even at the end of the game. Short and nested loops, optional and duplicated tasks are present as well[5].

5.2 Activity Prediction

The experimental procedure for the activity prediction task was as follows. First, each dataset was translated from its original representation to the input format of WoMan. Then, each dataset was split into training and test sets using a k-fold

[5] A loop consists of a piece, after a number of moves, going back to a square that it had occupied in the past. In short loops this happens within a few moves. Optional activities are involved in that a player might make the same move with or without taking another piece. Duplicate tasks are needed when a piece occupies the same square at different stages of the match, and thus in different contexts.

cross-validation procedure (see column *Folds* in Table 2 for the values of k: for GPItaly, 3 folds used due to the very large number of cases and activities in this dataset; for the others, 5 folds were used to provide the system with more learning information; both splits were used on Aruba to allow a comparison). Then, WIND (the learning functionality of WoMan) was used to learn models from the training sets. Finally, each model was used as a reference to call WEST and SNAP on each event in the test sets: the former checked compliance of the new event and suitably updated the set of statuses associated to the current case, while the latter used the resulting set of statuses to make a prediction about the next activity that is expected in that case. In these experiments, the predictions were based exclusively on the number of cases supporting each status.

Table 2. Prediction statistics

	Folds	Activity					Process					
		Pred	Recall	Rank	(Tasks)	Quality	Pos	(%)	C	A	U	W
Aruba	3	0.85	0.97	0.92	6.06	0.78	—	—	—	—	—	—
Aruba	5	0.87	0.97	0.91	5.76	0.78	—	—	—	—	—	—
GPItaly	3	0.99	0.97	0.96	8.02	0.92	—	—	—	—	—	—
black	5	0.42	0.98	1.0	11.8	0.42	2.06	(0.47)	0.20	0.00	0.15	0.66
white	5	0.55	0.97	1.0	11.27	0.54	1.60	(0.70)	0.44	0.00	0.15	0.40
draw	5	0.64	0.98	1.0	10.95	0.62	1.78	(0.61)	0.29	0.01	0.18	0.52
chess	5	0.54	0.98	1.0	11.34	0.53	1.81	(0.59)	0.31	0.00	0.16	0.53

Table 2 (section 'Activity') reports performance averages concerning the different processes (row 'chess' refers to the average of the chess sub-datasets). Column *Pred* reports in how many cases SNAP returned a prediction. Indeed, when tasks or transitions not present in the model are executed in the current enactment, WoMan assumes a new kind of process is enacted, and avoids making predictions. Among the cases in which WoMan makes a prediction, column *Recall* reports in how many of those predictions the correct activity (i.e., the activity that is actually carried out next) is present. Finally, column *Rank* reports how close it is to the first element of the ranking (1.0 meaning it is the first in the ranking, possibly with other activities, and 0.0 meaning it is the last in the ranking), and *Tasks* is the average length of the ranking. The *Quality* index is a mix of these values, obtained as

$$Quality = Pred \cdot Recall \cdot Rank \in [0, 1]$$

It is useful to have an immediate indication of the overall performance in activity prediction. When it is 0, it means that predictions are completely unreliable; when it is 1, it means that WoMan always makes a prediction, and that such a prediction is correct (i.e., the correct activity is at the top of the ranking).

First, note that, when WoMan makes a prediction, it is extremely reliable. The correct activity that will be performed next is almost always present in the

ranking (97–98 % of the times), and always in the top section (first 10 % items) of the ranking. For the chess processes it is always at the very top. This shows that WoMan is effective under very different conditions as regards the complexity of the models to be handled. In the Ambient Intelligence domain, this means that it may be worth spending some effort to prepare the environment in order to facilitate that activity, or to provide the user with suitable support for that activity. In the chess domain, this provides a first tool to make the machine able to play autonomously. The number of predictions is proportional to the number of tasks and transitions in the model. This was expected, because, the more variability in behaviors, the more likely it is that the test sets contain behaviors that were not present in the training sets. WoMan is almost always able to make a prediction in the Ambient Intelligence domain, which is extremely important in order to provide continuous support to the users. The percentage of predictions drops significantly in the chess domain, where however it still covers more than half of the match. Interestingly, albeit the evaluation metrics are different and not directly comparable, the *Quality* is at the same level or above the state-of-the-art performance obtained using Deep Learning [13] and Neural Networks [14]. The nice thing is that WoMan reaches this percentage by being able to distinguish cases in which it can make an extremely reliable prediction from cases in which it prefers not to make a prediction at all. The worst performance is on 'black', possibly because this dataset includes less training cases.

5.3 Process Prediction

The process prediction task was evaluated on the chess dataset, because it provided three different kinds of processes based on the same domain. So, we tried to predict the match outcome (white wins, black wins, or draw) as long as match events were provided to the system. The experimental procedure was similar to the procedure used for the activity prediction task. We used the same folds as for the other task, and the same models learned from the training sets. Then, we merged the test sets and proceeded as follows. On each event in the test set, WoGue was called using as candidate models *white*, *black*, and *draw*. In turn, it called WEST on each of these models to check compliance of that event and suitably update the corresponding sets of statuses associated to the current case. Finally, it ranked the models by increasing number of warnings in their 'best' status, where the 'best' status was the status that raised less warnings. Note that, in this case, WoMan always returns a prediction.

Table 2 (section 'Process') summarizes the performance on the process prediction task. Column *Pos* reports the average position of the correct prediction in the ranking (normalized in parentheses to $[0, 1]$, where 1 represents the top of the ranking, and 0 its bottom). The last columns report, on average, for what percentage of the case duration the prediction was correct (C: the correct process was alone at the top of the ranking), approximately correct (A: the correct process shared the top of the ranking with other, but not all, processes), undefined (U: all processes were ranked equal), or wrong (W: the correct process was not at the top of the ranking). All kinds of chess processes show the same

behavior. The overall percentage of correct predictions (C) is above 30 %, which is a good result, with A being almost null. Studying how the density of the different predictions is distributed along the match, we discovered that, on average, the typical sequence of outcomes is: *U-A-C-W*. This could be expected: total indecision happens at the beginning of the match (when all possibilities are still open), while wrong predictions are made towards the end (where it is likely that each single match has a unique final, never seen previously). In the middle of the game, where the information compiled in the model is still representative, correct or approximately correct predictions are more dense, which is encouraging. This also explains the high percentage of wrong predictions (53 %): since every match becomes somehow unique starting from 1/2–3/4 of the game, the learned model is unable to provide useful predictions in this phase (and indeed, predicting the outcome of a chess match is a really hard task also for humans). So, the percentage of correct predictions should be evaluated only on the middle phase of the match, where it is much higher than what is reported in Table 2.

6 Conclusions and Future Work

While traditionally exploited for checking process enactment conformance, process models may be used to predict which activities will be carried out next, or which of a set of candidate processes is actually being executed. The prediction performance may also provide clues about the correctness and reliability of the model. In some applications, such as Ambient Intelligence ones, there is more flexibility of behaviors than on industrial applications, which makes these predictions both more relevant and harder to be obtained. This paper proposes a way to make these kinds of predictions using the WoMan framework for workflow management. Experimental results on different tasks and domains suggest that the prediction ability of WoMan is noteworthy.

Given the positive results, we plan to carry out further work on this topic. First of all, we plan to check the prediction performance on other domains. Also, we will investigate how to use and mix different parameters to improve the prediction accuracy. Finally, we would like to embed the prediction module in other applications, in order to guide their behavior.

Acknowledgments. Thanks to Amedeo Cesta and Gabriella Cortellessa for providing the GPItaly dataset, and to Riccardo De Benedictis for translating it into WoMan format. This work was partially funded by the Italian PON 2007-2013 project PON02_00563_3489339 'Puglia@Service'.

References

1. Weijters, A., van der Aalst, W.: Rediscovering workflow models from event-based data. In: Hoste, V., Pauw, G.D. (eds.) Proceedings of the 11th Dutch-Belgian Conference of Machine Learning (Benelearn 2001), pp. 93–100 (2001)

2. Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28108-2_19
3. Ferilli, S., Esposito, F.: A logic framework for incremental learning of process models. Fund. Inform. **128**, 413–443 (2013)
4. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Alonso, G., Saltor, F., Ramos, I. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 467–483. Springer, Heidelberg (1998)
5. Cook, J., Wolf, A.: Discovering models of software processes from event-based data. Technical report CU-CS-819-96, Department of Computer Science, University of Colorado (1996)
6. van der Aalst, W.: The application of petri nets to workflow management. J. Circ. Syst. Comput. **8**, 21–66 (1998)
7. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**, 1128–1142 (2004)
8. Ferilli, S.: Woman: Logic-based workflow learning and management. IEEE Trans. Syst. Man Cybern. Syst. **44**, 744–756 (2014)
9. Pesic, M., van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In: Eder, J., Dustdar, S. (eds.) BPM 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
10. Muggleton, S.: Inductive logic programming. New Gener. Comput. **8**, 295–318 (1991)
11. Herbst, J., Karagiannis, D.: An inductive approach to the acquisition and adaptation of workflow models. In: Proceedings of the IJCAI 1999 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business, pp. 52–57 (1999)
12. Coradeschi, S., Cesta, A., Cortellessa, G., Coraci, L., Gonzalez, J., Karlsson, L., Furfari, F., Loutfi, A., Orlandini, A., Palumbo, F., Pecora, F., von Rump, S., Štimec, Ullberg, J., tslund, B.: Giraffplus: Combining social interaction and long term monitoring for promoting independent living. In: Proceedings of the 6th International Conference on Human System Interaction (HSI), pp. 578–585. IEEE (2013)
13. Lai, M.: Giraffe: using deep reinforcement learning to play chess. CoRR abs/1509.01549 (2015)
14. Oshri, B., Khandwala, N.: Predicting moves in chess using convolutional neural networks. In: Stanford University Course Project Reports - CS231n: Convolutional Neural Networks for Visual Recognition (Winter 2016)