# Extended Process Models for Activity Prediction

**Conference Paper** · June 2017

DOI: 10.1007/978-3-319-60438-1_36

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

# Extended Process Models for Activity Prediction

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

[1] Dipartimento di Informatica – Università di Bari, Bari, Italy
{stefano.ferilli, floriana.esposito, sergio.angelastro}@uniba.it
[2] 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 used to predict which activities will be carried out next. The prediction performance may provide indirect indications on the correctness and reliability of a process model. This paper proposes a strategy for activity prediction using the WoMan framework for workflow management. It extends a previous approach, that has proved to be able to handle complex processes. Experimental results on different domains show an increase in prediction performance compared to the previous approach.

**Keywords:** Process Mining, Activity Prediction, Process Model

## 1 Introduction & Background

A *process* consists of actions performed by agents [1, 2]. A *workflow* is a formal specification of a process. It may involve sequential, parallel, conditional, or iterative execution [12]. A process execution, compliant to a given workflow, is called a *case*. It can be described as a list of *events* (i.e., identifiable, instantaneous actions, including decisions upon the next activity to be performed), associated to *steps* (time points) and collected in *traces* [13]. Relevant events are the start and end of process executions, or of activities [2]. 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).

Process Management techniques are useful in domains where a production process must be monitored (e.g. in the industry) in order to check whether the actual behavior is compliant with a desired one. When a process model is available, new process enactments can be automatically supervised. The complexity of some domains requires to learn automatically the process models, because building them manually would be very complex, costly and error-prone. Process Mining [14, 9] approaches aim at solving this problem. *Declarative* process mining approaches learn models expressed as a set of constraints, instead of a monolithic model (usually expressed as some kind of graph) [11].

The WoMan framework [6, 4] lies at the intersection between *Declarative* Process Mining and Inductive Logic Programming (ILP) [10]. Indeed, it pervasively uses 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 WoMan can handle efficiently and effectively very complex processes, thanks to its powerful representation formalism and process handling operators. 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 allows to carry out continuous adaptation of the learned model to the actual practice efficiently, effectively and transparently to the users [4].

A relevant issue in Process Management in general, and in Process Mining in particular, is to assess how well can a model provide hints about what is going on in the process execution, and what will happen next. Indeed, 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. The task of activity prediction may be stated as follows: given a process model and the current (partial) status of a new process execution, guess which will be the next activity that will take place in the execution. In industrial environments, the rules that determine how the process must be carried out are quite strict; so, predicting the process evolutions is a trivial consequence of conformance checking. Other, less traditional application domains (e.g., the daily routines of people at home or at work, seen as a process), involve much more variability, and obtaining reliable predictions becomes both more difficult and more useful. Another, very relevant and interesting, application of process-related predictions is in the assessment of the quality of a model. Indeed, since models are learned automatically exactly because the correct model is not available, only an empirical validation can be run. In literature, this is typically done by applying the learned model to new process enactments.

In particular, this paper focuses on the approach adopted by WoMan [7] to carry out the activity prediction task, given its good results in various domains. We propose here two new contributions: first, we extend the Woman's process model with additional information aimed at improving the prediction performance; second, we report for the first time the detailed prediction algorithm. The next two sections present WoMan, its (extended) formalism and its approach to activity prediction. Then, Section 4 reports and comments about the experimental outcomes. Finally, in the last section, we draw some conclusions and outline future work issues.

## 2 The WoMan Formalism

WoMan representations are based on the Logic Programming formalism, and works in Datalog, where only constants or variables are allowed as terms. Following foundational literature [1, 8], trace elements in WoMan are 6-tuples, represented in WoMan as facts `entry(`$T$`,`$E$`,`$W$`,`$P$`,`$A$`,`$O$`).`, that report information about relevant events for the case they refer to. $T$ is the event timestamp, $E$ is the type of the event, $W$ is the name of the workflow the process refers to, $P$ is a unique identifier for each process execution, $A$ is the name of the activity, and $O$ is the progressive number of occurrence of that activity in that process. An optional field, $R$, can be added to specify the agent that carries out activity $A$. In particular, $E$ is one of $\{\mathbf{begin}|\mathbf{end}\}\_\{\mathbf{process}|\mathbf{activity}\}$. Activity begin and end events are needed to properly handle time span and parallelism of tasks [13]. Since parallelism among activities is explicit, there is no need for inferring it by means of statistical (possibly wrong) considerations. In each case, the activities are uniquely identified by a progressive number called *step*. $E =$ **context_description** is used to describe contextual information at time $T$, in the form of a conjunction of FOL atoms built on domain-specific predicates.

WoMan models are expressed as sets of atoms built on different predicates. The predicates used in previous versions:

- `task(`$t$`,`$C$`)` : task $t$ occurred in training cases $C$.
- `transition(`$I$`,`$O$`,`$p$`,`$C$`)` : transition[3] $p$, occurred in training cases $C$, is enabled if all input tasks in $I = [t'_1, \ldots, t'_n]$ are active; if fired, after stopping the execution of all tasks in $I$ (in any order), the execution of all output tasks in $O = [t''_1, \ldots, t''_m]$ is started (again, in any order). If several instances of a task can be active at the same time, $I$ and $O$ are multisets, and application of a transition consists in closing as many instances of active tasks as specified in $I$ and in opening as many activations of new tasks as specified in $O$.
- `task_agent(`$t$`,`$A$`)` : an agent, matching the roles $A$, can carry out task $t$.
- `transition_agent(`$[a'_1, \ldots, a'_n]$`,`$[a''_1, \ldots, a''_m]$`,`$p$`,`$C$`,`$q$`)` : transition $p$, involving input tasks $I = [t'_1, \ldots, t'_n]$ and output tasks $O = [t''_1, \ldots, t''_m]$, may occur provided that each task $t'_i \in I, i = 1, \ldots, n$ is carried out by an agent matching role $a'_i$, and that each task $t''_j \in O, j = 1, \ldots, m$ is carried out by an agent matching role $a''_j$; several combinations can be allowed, numbered by progressive $q$, each encountered in cases $C$.

The core of the model is expressed by `task/2` and `transition/4`. The latter, in particular, represent the allowed connections between activities in a very modular way. This allows WoMan to check if a new execution corresponds to at least one training case, or if it can be obtained by mixing transitions taken from different training cases. Since a task or transition $t$ may occur many times in the same case, $C$ is defined as a multiset. It allows WoMan to compute the probability of $t$ by means of its relative frequency in cases (as shown in [4]). In this way, WoMan can ignore less frequent $t's$, since they could be noisy. Also, the multiplicity of a case in $C$ allows to set a limit on the number of repetitions of $t$ during an execution (avoiding longer loops than expected). In this work, the set of predicates was extended to express *temporal constraints* on the activities:

[3] Note that this is a different meaning than in Petri Nets.

- `task_time(`$t$`,`$[b'$`,`$b'']$`,`$[e'$`,`$e'']$`,`$d$`)` : task $t$ must begin at a time $i_b \in [b', b'']$ and end at a time $i_e \in [e', e'']$, and has average duration $d$;
- `transition_time(`$p$`,`$[b', b'']$`,`$[e', e'']$`,`$g$`,`$d$`)` : transition $p$ must begin at a time $i_b \in [b', b'']$ and end at a time $i_e \in [e', e'']$; it has average duration $d$ and requires an average time gap $g$ between the end of the last input task in $I$ and the activation of the first output task in $O$;
- `task_in_transition_time(`$t$`,`$p$`,`$[b', b'']$`,`$[e'$`,`$e'']$`,`$d$`)` : task $t$, when run in transition $p$, must begin at a time $i_b \in [b', b'']$ and end at a time $i_e \in [e', e'']$, and has average duration $d$;
- `task_step(`$t$`,`$[b', b'']$`,`$[e', e'']$`,`$d$`)` : task $t$ must start at a step $s_b \in [b', b'']$ and end at a step $s_e \in [e', e'']$, along an average number of steps $d$;
- `transition_step(`$p$`,`$[b', b'']$`,`$[e', e'']$`,`$g$`,`$d$`)` : transition $t$ must start at a step $s_b \in [b', b'']$ and end at a step $s_e \in [e', e'']$, along an average number of steps $d$ and requires an average gap of steps $g$ between the end of the last input task in $I$ and the activation of the first output task in $O$;
- `task_in_transition_step(`$t$`,`$p$`,`$[b', b'']$`,`$[e', e'']$`,`$d$`)` : task $t$, when run in transition $p$, must start at a step $s_b \in [b', b'']$ and end at a step $s_e \in [e', e'']$, along an average number of steps $d$.

These temporal constraints are mined on the entire training set. Begin and end time are relative to the start of the process execution, computed as the timestamp difference between the begin of process and the event they refer to. Step information is computed on the progressive number $s$ on which a task $t$ is executed. Consider, for instance, task *act_Meal_Preparation* and transition $p23 : \{act\_Meal\_Preparation\} \Rightarrow \{act\_Meal\_Preparation, act\_Relax\}$; an example of the new components for them might be:

`task_time(`*act_Meal_Preparation*`,[5,10],[10,21],8.62)`
`transition_time(`$p23$`,[6,8],[20,25],17.34,10.12)`
`task_step(`*act_Meal_Preparation*`,[s3,s5],[s7,s12],2.5)`
`transition_step(`$p23$`,[s4,s5],[s10,s13],8.3,3.3)`
`task_in_trans_step(`*act_Meal_Preparation*`,`$p23$`,[s10,s11],[s11,s12],1.4)`
`task_in_trans_time(`*act_Meal_Preparation*`,`$p23$`,[10,12],[18,21],8.3)`

Finally, WoMan can expresses pre/post conditions (that specify what must be true for executing a given task, transition or a task in the context of a specific transition) as FOL rules based on contextual and control flow information. Conditions are not limited to the current status of execution. They may involve the status at several steps using two predicates:

- `activity(`$s$`,`$t$`)` : at step $s$ (unique identifier) $t$ is executed;
- `after(`$s'$`,`$s''$`,[`$n'$`,`$n''$`],[`$m'$`,`$m''$`])` : step $s''$ follows step $s'$ after a number of steps ranging between $n'$ and $n''$ and after a time ranging between $m'$ and $m''$.

Due to concurrency, predicate `after/3` induces a partial ordering on the set of steps.

## 3 Workflow Prediction

WoMan's supervision module, **WEST** (Workflow Enactment Supervisor and Trainer) [7], checks whether new cases are compliant with a given model, returning suitable warnings (deviations from model) or errors (case trace syntactically wrong) in case it is not. In [7], warnings concerned unexpected tasks or transitions, conditions not fulfilled, unexpected resources running a given activity. Here, additional warnings were considered, expressing deviations from the intervals specified by the new constraints on time and steps introduced in the model. Each kind of warning was associated to a numerical weight (currently determined heuristically) that quantifies the associated degree of severity. As explained in [7], in WoMan a partial process execution might be compliant to several different exploitations of the model. To handle this ambiguity, WEST maintains the set $S$ of these exploitations, each of which is called *status* and represented as a 5-tuple of sets $\langle M, R, C, T, W\rangle$ recording the following information:

***M*** the marking, i.e., terminated activities, not yet used to fire a transition;
***R*** (for 'Ready') the output activities of fired transitions in the status, and that the system is waiting for in order to complete them;
***C*** training cases that are compliant with that status;
***T*** (hypothesized) transitions that have been fired to reach that status;
***W*** multiset of warnings raised by the various events that led to that status.

As long as the process executions proceeds, new alternative statuses may be added to $S$, and statuses that are not compliant with the model may be removed. For each surviving status, its *discrepancy* from the model, $\delta(status)$, can be computed based on $W$, to be used in the prediction procedure. Compared to [7], here it takes into account also the execution of a task or transition being not consistent with the learned time constraints. Since each *status* may be associated with different activities to be performed next, there is also an ambiguity about which activities that will be carried out. The activity prediction module of WoMan, **SNAP** (Suggester of Next Action in Process), exploits $S$ (maintained by WEST) to compute statistics that are useful to determine which are the expected next activities and to rank them by some sort of likelihood, according to Algorithm 1. In a preliminary phase each $status \in S$ is removed and evaluated. For each transition in the model enabled by the *Marking* component of *status*, the evolution $status'$ of *status* is added to $S$, and $W$ is updated if inconsistency from observed and learned behavior is encountered. Then, the discrepancy $\delta(status)$ of each $status \in S$ is measured, and statuses that exceed a certain discrepancy tolerance threshold are removed. Based on the statuses with a low discrepancy only, the set *Nexts* of actions/tasks that may be carried out next is selected from the *Ready* component of each $status \in S$, and, finally, actions in *Nexts* are scored and ranked based on a heuristic combination of the following parameters, computed over $S$:

1. $\mu(a)$, multiplicity of $a$ across the various statuses (activities that appear in more statuses are more likely to be carried out next);

**Algorithm 1** Prediction of possible next actions in SNAP

```
Require: $\mathcal{M}$: process model
Require: $Statuses$ : set of currently compliant statuses compatible with the case
Require: $Event$ : current event of trace
Require: $\epsilon$: a threshold to filter only more compliant statuses
  if $E$ = end_activity $\vee$ $E$ = begin_process then
    for all $S = \langle M, R, C, T, W \rangle \in Statuses$ do
      $Statuses \leftarrow Statuses \setminus \{S\}$
      for all $p : I \Rightarrow O \in \mathcal{M}$ do
        if $I \subseteq Marking$ then
          $C' \leftarrow C \cap C_p$ /* $C_p$ training cases involving $p$ */
          $W' \leftarrow W \cup W_{A,p,S}$ /* $W_{A,p,S}$ warnings raised by running $A$ in $p$ given $S$ */
          $Statuses \leftarrow Statuses \cup \{\langle M \setminus I, R \cup O, C', T\&\langle t \rangle, W' \rangle\}$
    $NewStatuses \leftarrow Statuses \setminus \{S\}$
  if $E$ = begin_activity then
    $NewStatuses \leftarrow Statuses$
  $Nexts = \{a \mid \forall S = \langle M, R, C, T, W \rangle \in NewStatuses : discrepancy(S) > \epsilon \wedge a \in R\}$
  /* Nexts is the multiset of candidate next actions */
  $Ranking \leftarrow \{\}$
  for all $a \in Nexts$ do
    $StatusWithA = \{\langle M, R, C, T, W \rangle \in NewStatuses \mid a \in R\}$
    $\delta_a = \sum_{S \in StatusWithA} \delta(S)$ /* summation of status's $\delta$ involving $a$ */
    $C_a = \bigcup_{\langle M,R,C,T,W \rangle \in StatusWithA} C$ /* union of status's cases involving $a$ */
    $score \leftarrow (\mid C_a \mid \cdot \mid StatusWithA \mid \cdot \mu(a)) \ / \ \delta_a$
    $Ranking \leftarrow Ranking \cup \{\langle score, a \rangle\}$
```

2. $C_{status}$, 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. $\delta(status)$, sum of weights of warnings raised by the status in which the action is included (activities expected in statuses that raised less warnings are more likely to be carried out next).

## 4 Evaluation

The performance of the proposed activity prediction approach was evaluated on several datasets, concerning different kinds of processes associated with different kinds and levels of complexity. The datasets related to Ambient Intelligence concern typical user behavior. Thus, they involve much more variability and subjectivity than in industrial process, and there is no 'correct' underlying model, just some kind of 'typicality' can be expected:

**Aruba** from the CASAS benchmark repository[4]. It includes continuous recordings of home activities of an elderly person, visited from time to time by

[4] http://ailab.wsu.edu/casas/datasets.html

**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 |

her children, in a time span of 220 days. Each day is mapped to a case of the process representing the daily routine of the elderly person. Transitions correspond to terminating some activities and starting new activities. The resources (persons) that perform activities are unknown.

**GPItaly** from one of the Italian use cases of the GiraffPlus project[5] [3]. It concerns the movements of an elderly person (and occasionally other people) in the various rooms of her home along 253 days. Each day is a case of the process representing the typical movements of people in the home. Tasks correspond to rooms; transitions correspond to leaving a room and entering another.

The other concerns chess playing, where again the 'correct' model is not available:

**Chess** from the Italian Chess Federation website[6]. 400 reports of actual top-level matches were downloaded. Each match is a case, belonging to one of 3 processes associated to the possible match outcomes: *white* wins, *black* wins, or *draw*. A task is the occupation of a square by a specific kind of piece (e.g., "black rook in a8"). Transitions correspond to moves: 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). The involved resources are the two players: 'white' and 'black'.

Table 1 reports statistics on the experimental datasets: number of cases and number of events, activities, tasks and transitions, also on average per case. 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. The datasets are different also from a qualitative viewpoint. Aruba cases feature many short loops and some concurrency (involving up to 2 activities), optional and duplicated activities. The same holds for GPItaly, except for concurrency. The chess datasets are characterized by very high concurrency: each game starts with 32 concurrent activities (a number which is beyond the reach of many current process mining systems [6]). This number progressively decreases (but remains still high) as long

[5] `http://www.giraffplus.eu`
[6] `http://scacchi.qnet.it`

**Table 2.** Prediction statistics

| new (old) | Folds | Activity Prediction | | | | |
|---|---|---|---|---|---|---|
| | in [7] | Pred | Recall | Rank | Tasks | Quality |
| Aruba | 3 | 0.88 (+0.03) | 0.97 (=) | 0.86 (-0.06) | 6.3 (+0.24) | 0.78 (=) |
| GPItaly | 3 | 1.0 (+0.01) | 0.99 (+0.02) | 0.98 (+0.02 ) | 8.2 (+0.28) | 0.97 (+0.05) |
| black | 5 | 0.53 (+0.11) | 0.98 (=) | 1.0 (=) | 11.09 (-0.71) | 0.51 (+0.09) |
| white | 5 | 0.55 (=) | 0.98 (+0.01) | 1.0 (=) | 10.9 (-0.37) | 0.5 (+0.01) |
| draw | 5 | 0.65 (+0.01) | 0.98 (=) | 1.0 (=) | 10.6 (-0.35) | 0.64 (+0.02) |
| *chess* | 5 | 0.58 (+0.04) | 0.98 (=) | 1.0 (=) | 10.90 (-0.44) | 0.55 (+0.02) |

as the game proceeds. Short and nested loops, optional and duplicated tasks are present as well. The number of agent and temporal constraints is not shown, since the former is at least equal, and the latter is exactly equal, to the number of tasks and transitions.

The experimental procedure was as follows. First, each dataset was translated from its original representation to the input format of WoMan. Then, a 10-fold cross-validation procedure was run for each dataset, using the learning functionality of WoMan (see [5]) to learn models for all 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 (as described in the previous section).

Table 2 reports average performance for the processes on the row headings ('chess' refers to the average of the chess sub-datasets). Column *Pred* reports the ratio of cases in which 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. Column *Recall* reports the ratio of cases in which the correct activity (i.e., the activity that is actually carried out next) is present in the ranking, among those in which a prediction was made. 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, and 0.0 meaning it is the last in the ranking), and *Tasks* is the average length of the ranking (the lower, the better). $Quality = Pred \cdot Recall \cdot Rank \in [0,1]$ is a global index that provides an immediate indication of the overall activity prediction performance. 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). Since the proposed approach extends the one in [7], a reference to the differences with respect to its outcomes is also reported in parentheses, for each column in Table 2 ( '+' indicating an improvement, '−' a degradation, and '=' equal performance). Column *Folds* shows the number of folds used for the previous $k$-fold cross validation, since the current one runs the 10-fold for each dataset.

First, note that *Quality* index in the current approach outperforms the previous in [7]. This is due to the WoMan's ability to ensure more reliable predictive support, as the model includes both more constraints and cases. Specifically, *Rank* index is improved in all cases, except on Aruba. However, this decrease is balanced by improved Pred, which leaves the *Quality* unchanged. This is acceptable, because, while WoMan can make a prediction in more cases, a 6% decrease in ranking over 6.3 tasks means a loss of much less than 1 position (0.378).

WoMan is extremely reliable because the correct next activity is almost always present in the ranking (97-99% of the times), and always in the top section (first 10% items) of it, especially for the chess processes (always at the top). Compared to previous one, current *Recall* and *Rank* tend to be equal, except for GPItaly. The former is because of different fold setting, while the latter probably due to the temporal constraints. This confirm 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. Compared to previous results, the number of predictions, in all domains, is increased, due to the fact that the 10-fold provide a bigger training, involving more information. 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. While neatly lower, the percentage of predictions in the chess domain was clearly improved, and covers more than half of the match. 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.

## 5 Conclusions

In addition to other classical exploitations, process models may be used to predict the next activities that will take place. This would allow to take suitable actions to help accomplishing those activities. This paper proposed an extended approach to make these kinds of predictions using the WoMan framework for workflow management. Experimental results on different tasks and domains suggest that the proposed approach can successfully perform such predictions.

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, e.g. Industry 4.0 ones. Also, we will investigate how to improve the prediction accuracy by means of more refined strategies. Finally, we would like to embed the prediction module in other applications, in order to guide their behavior.

## References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, 1998.
2. J.E. Cook and A.L. Wolf. Discovering models of software processes from event-based data. Technical Report CU-CS-819-96, Department of Computer Science, University of Colorado, 1996.
3. S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, J. Gonzalez, L. Karlsson, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo, F. Pecora, S. von Rump, Štimec, J. Ullberg, and B. tslund. Giraffplus: Combining social interaction and long term monitoring for promoting independent living. In *Proc. of the 6th International Conference on Human System Interaction (HSI)*, pages 578–585. IEEE, 2013.
4. S. Ferilli. Woman: Logic-based workflow learning and management. *IEEE Transaction on Systems, Man and Cybernetics: Systems*, 44:744–756, 2014.
5. S. Ferilli. WoMan: Logic-based workflow learning and management. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44:744–756, 2014.
6. S. Ferilli and F. Esposito. A logic framework for incremental learning of process models. *Fundamenta Informaticae*, 128:413–443, 2013.
7. Stefano Ferilli, Floriana Esposito, Domenico Redavid, and Sergio Angelastro. Predicting process behavior in woman. In *AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings*, pages 308–320, 2016.
8. J. Herbst and D. Karagiannis. An inductive approach to the acquisition and adaptation of workflow models. In *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 52–57, 1999.
9. IEEE Task Force on Process Mining. Process mining manifesto. In *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. 2012.
10. S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.
11. M. Pesic and W. M. P. van der Aalst. A declarative approach for flexible business processes management. In *Proceedings of the 2006 international conference on Business Process Management Workshops*, BPM'06, pages 169–180. Springer-Verlag, 2006.
12. W.M.P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8:21–66, 1998.
13. W.M.P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16:1128–1142, 2004.
14. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering workflow models from event-based data. In *Proc. 11th Dutch-Belgian Conference of Machine Learning (Benelearn 2001)*, pages 93–100, 2001.