Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 306–315

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

PrOnto: an Ontology Driven Business Process Mining Tool

Stefano Bistarelli[a], Tommaso Di Noia[b], Marina Mongiello[b], Francesco Nocera[b,*]

[a] *Universitá di Perugia, Dipartimento di Matematica e Informatica, Via Vanvitelli n.1, Perugia, 06123, Italy*
[b] *Politecnico di Bari, Dipartimento di Ingegneria Elettrica e dell'Informazione, Via E. Orabona n. 4, Bari, 70125, Italy*

Abstract

The main aim of data mining techniques and tools is that of identify and extract, from a set of (big) data, implicit patterns which can describe static or dynamic phenomena. Among these latter business processes are gaining more and more attention due to their crucial role in modern organizations and enterprises. Being able to identify and model processes inside organizations is for sure a key asset to discover their weak and strong points thus helping them in the improvement of their competitiveness. In this paper we describe a prototype system able to discover business processes from an event log and classify them with a suitable level of abstraction with reference to a related business ontology. The identified process, and its corresponding level of abstraction, depends on the knowledge encoded in the reference ontology which is dynamically exploited at runtime. The tool has been validated by considering examples and case studies from the literature on process mining.

Peer-review under responsibility of KES International

Keywords: business process mining, ontology, activity diagram

1. Introduction and motivation

Nowadays, modern organizations and enterprises have recognized the key role played by business process engineering and monitoring in their race towards competitiveness in more and more crowded markets. Improvement in business processes would ensure organizations and enterprises to be more competitive and able to better check profit margins and revenues. Business processes identification and monitoring may help in reaching this goal by supporting more precise and optimized decision making strategies. A possible solution is that of performing process mining [1] in stored data in order to get knowledge about processes. In this respect, event logs and information about resources, activities and transitions from states in a workflow modeling manner might be mined to extract process description. In this paper we present our prototype tool – PrOnto – able to analyze and extract knowledge on business processes in order to discover patterns, structures and operations within event logs. PrOnto has been conceived as to support the organizational decision making. In its operations, the tool exploits the knowledge encoded in an ontology domain to model the description of the process it provides as output with different levels of abstraction. Our approach might seem similar to that used in the paper [2], anyway instead of using an ontology of actions, we developed an ontology

* Corresponding author. Tel.: +0805963641 ; fax: +0805963410.
E-mail address: francesco.nocera@poliba.it

1877-0509
Peer-review under responsibility of KES International
10.1016/j.procs.2017.08.002

that represents the organization, hence we make abstraction on the person performing the action rather than on the action. This allows us to consider flows of activites and to map them in any integrity policy.

Thus the knowledge modeled in our ontology is more similar to the model proposed in the paper by T. Finin et al.[3]: the process allows to check if the company has implemented any correct Role-Based Access Control (RBAC) access control policy.

With respect to existing work in literature, in particular one of the more relevant[4], PrOnto has two main advantages: first of all it produces a UML Activity Diagram (AD) in XML format. The use of XMI Metadata Interchange (XMI)[5] format for representing diagram and makes ease the translation in other formalisms or languages. Secondly, it that takes in account the enterprise (hierarchical) organization and uses a knowledge based approach, by means of an ontology to model information extracted from the event log. To sum up: PrOnto takes as input an event log file in comma-separated values (CSV) format and produces an UML-based AD representing the business process generating the input log. ADs are powerful tools as they combine ideas from several techniques: the event diagrams, Specification and Description Language (SDL) state modeling techniques[6], workflow modeling[7], and Petri Nets. The ontology allows to model of the description of the process it provides as output and consider them in building the process model with different levels of abstraction depending on the user's preference.

The chosen encoding for the AD is a simplified version of XML Metadata Interchange (XMI)[5] tailored to our process discovery needs. The proposed tool is then featured as an ontology-driven system able both to take into account the resource that executes the action and to describe the related business process at different level of resource abstraction.

The remaining part of the paper is organized as follows. In Section 2 we present PrOnto together with the algorithm and the formalisms it exploits. The tool is presented with the aid of a running example. In Section 4 we describe state of the art related works and compare them with our approach. Conclusions and future work close the paper.

2. Background

The extraction of a workflow starting from an event log is a task which is typically performed via process mining[1], in 1998 to study to workflows[8]. Via process mining, it is possible to collect logs of events in order to automatically construct models of the behavior stored in the events log. Logs represent the history of a systems and they typically contain: (i) an *activit*y, (task or operation); (ii) a *case*, that is the instance of the process – the object that is to be handled: a job application, a building permit, a request check, etc; (iii) a *resource*, that is the performer of the activity.

The *Alpha algorithm*[9] is the first algorithm able to capture concurrency in business processes extracted from an input log file. It processes sequences of temporal activities (traces) ranked for each case present in an event log. Examples of traces are: $T1 = \{a, b, c, d\}$, $T2 = \{a, c, b, d\}$, $T3 = \{a, c, d\}$. The *alpha algorithm* identifies causality between activities, based on four types of ordering relations: (1) *direct sequence* is the temporal order; (2) *direction of causality* between activity; (3) the potential *parallelism of the activities*; (4) the activities that *do not follow one another directly*. Alpha algorithm was proved to be correct for a large class of processes, despite having problems of noise and incompleteness.

A trace is a sequence of temporal events ranked for each case present in an event log. Examples of traces are: $T_1 = \{a, b, c, d\}, T2 = \{a, c, b, d\}, T3 = \{a, c, d\}$. The alpha algorithm identifies causality between activities, based on four types of ordering relations. The first relation (direct sequence) is the temporal order. The second relation is the direction of causality between activity. The third relation represents the potential parallelism of the activity. The fourth and final relation represents the activities that do not follow one another directly. The stated rules are checked according to the following steps:

- Step 1: check that the activity does not appear in the register of the transitions generated by Workflow Net.
- Step 2: identify the set of the starting activity, that is, all the activity that appear in the first position within a track.
- Step 3: identify the set of end of activity, that is, all the activity that appear in the last position within a track.
- Step 4 and 5 are the core of the alpha Algorithm: in these steps the algorithm determines the places of a Workflow Net and their connections. The construction of places 1, $p(A, B)$ such that A is the set of input transitions and B the set of output transitions, all the elements of A should have causal dependencies with all

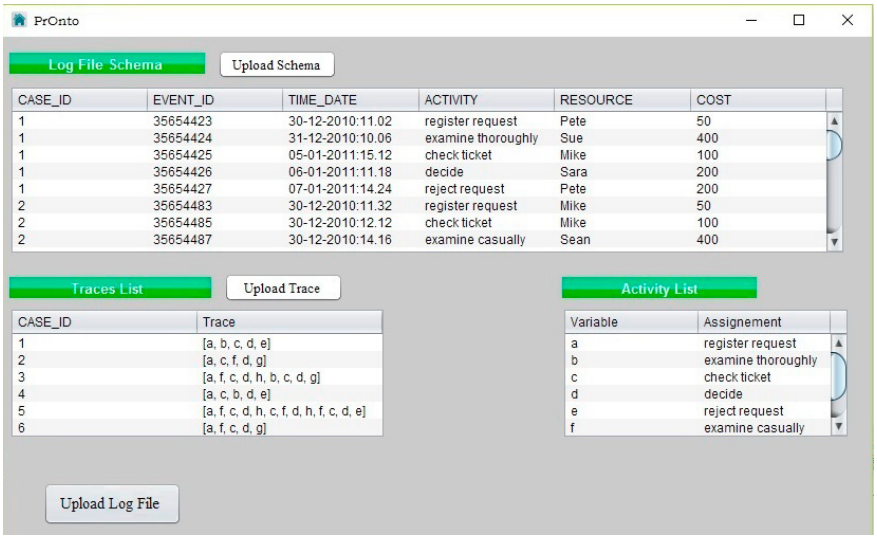

Fig. 1. Screenshot of the tool execution.

the elements of B; Furthermore, the elements of A must not come after A and similar requirement applies to B. These constraints on the elements of A and B enable the correct extraction of the constructs of AND-split / join and OR-split / join. Note that the OR-split / join requires the fusion of places.

- Step 6: refine the exact amount of places discovered on the net.
- Step 7: create places.
- Step 8: the places are connected to the respective input / output transitions.
- Step 9: The Workflow Net is built.

3. Process monitoring and analysis with the aid of Ontologies using PrOnto

3.1. Formal framework

PrOnto has been conceived as an ontology-driven support for process monitoring and analysis. PrOnto [1] was developed in Java, the development environment is Eclipse Version Neon 1, an open source tool. The platform on which proofs have been executed is a Laptop with 8 GB of RAM, Intel I7 processor 2.3 GHz. Data read from the log files were managed by a Database accessed through a MySql Connector Library to be included within Eclipse version Connector / J 5.1.40 and MySql Server version 5.7.17. The Tool supports as minimum requirements a 512 Mb Ram and Pentium 4 2.5 GHz; maximum storage space 50 Mb.

The input of the tool is an event log text-file in CSV format, whereas the output is an XMI file (adopting a simplified syntax) to formalize Activity Diagram mapped on the extracted process. The tool implements *alpha algorithm*[9], but differently from the original version it builds an AD instead of a Petri net, and abstracts the process resources.

Fig. 1 shows the GUI of the tool with the details of an example event log file. Information about the process has been modeled in a knowledge base. More specifically, we have developed an OWL 2 ontology[2] to represent both resources and activities by using *Protégé Version 5.1.0*[3] editor. The ontology is not hardcoded within the tool but it is loaded at runtime through a parametric link. Fig. 2 shows the class hierarchy graph of our ontology.

[1] PrOnto's web page is available online at http://sisinflab.poliba.it/index.php?page=tools.
[2] `https://www.w3.org/TR/owl2-overview/`
[3] `http://protege.stanford.edu/`

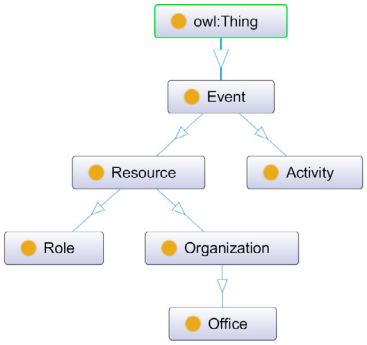

Fig. 2. Class hierarchy graph of our ontology.

Activities are the tasks to be performed composing a process. They are assumed to be elementary, and therefore have a precise and immediate translation in specific tasks. The processes are instead defined as a composition of activities. Resources represent agents or roles related to the activities in the process. A resource is the process performer: generally an executor can be a person, an object or a particular software program or the execution could take several performers of different types at the same time. In particular, within corporate and administrative structures, we may raise the problem of identifying the person in charge of performing a certain part of a specific process. It is noteworthy that the assignment of the performers to the activities can be carried out flexibly and in order to minimize the changes needed in front of changes in the structure of the process or of the organization. We model the assignment of the resources to the activity by abstraction on the Ontology. There is an abstraction level that represents the absence of resource encoding in the activity performance, whereas a lower level of abstraction means that activities are distinguished based on resources. The abstraction level is given as input to the tool before generating the output (UML AD). Let us indicate with AL the level of abstraction given as input to the tool with $AL \in \{0, 1, \ldots\}$ where $AL = k$ depends on the depth within the ontology of the class mapping the resources. Suppose that the tool receives as input $AL = 0$, this means that there is total abstraction on resources; the tool manages just activities, otherwise – with a lower level of abstraction – the tool distinguishes activities based on resources.

3.2. Use case scenario

To validate the proposed approach, let us consider a toy example meaningful to show the two relevant aspects of the tool with respect to existing work: the output format as an AD and the different levels of the modeled ontology depending on the users preference.

To validate the proposed appraoch, let us consider a small example describing The scenario starts from the event log[4] shown in Fig. 1. The event log is a record that shows the case, the identifier of the event, the timestamp, the name of the activity, the resource that executes the activity and the cost of executing the activity. The screenshot shows a part of the log of events, for sake of simplicity, the screenshot shows just 8 events out of 43 events in the log made up of 8 activities, 6 resources and 6 cases. The event log describes the following activities: `register request, examine thoroughly, check ticket, decide, reject request, register request, examine casually, pay compensation, reinitiate request`.
Cases are numbered from 1 to 6 and resources are: `Pete, Mike, Sue, Sara, Ellen, Sean`.
An identifier is assigned to each activity, as follows:
`a = register request, b = examine thoroughly, c = examine casually, d = check ticket, e = decide, f = reject request, g = pay compensation, h = reinitiate request.`
This modeling refers to the case of high abstraction, hence the activities are not distinguished based on the resources. The tool executes the modified version of the alpha algorithm and extracts the following traces from the log:

[4] Taken from the website `http://www.processmining.org/event_logs_and_models_used_in_book`

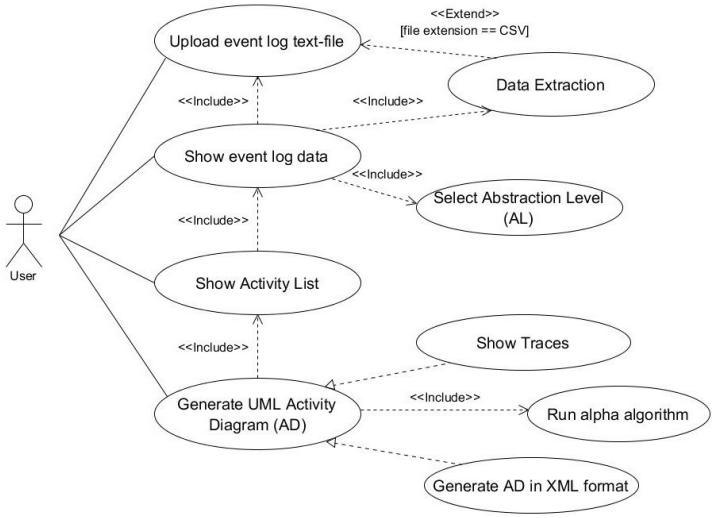

Fig. 3. Use case model of PrOnto.

```
case 1 (a, b, c, d, e)
case 2 (a, c, f, d, g)
case 3 (a, f, c, d, h, b, c, d, g)
case 4 (a, c, b, d, e)
case 5 (a, f, c, d, h, c, f, d, h, f, c, d, e)
case 6 (a, f, c, d, g)
```

Suppose the user requires a high level of abstraction, if `Pete` and `Mike` work in the same office, this means that $AL = 0$ hence the tool does not distinguish between activities executed by `Pete` or by `Mike`; so in the activity diagram there is just one layer to which the activities are mapped, this means for example that there is just one activity `register request`. On the other hand, suppoer that the user needs a lower abstraction level, more specifically $AL = 1$, hence the tool would distinguish as different activities those executed by `Pete` and by `Mike` whenever they occur, hence the activity `register request` is considered a different activity depending on the performer (resource). The ontology is represented as a dependency graph mappable on the BPMN (Business Process Modeling Notation) or on the AD; by varying the level in the abstract ontology both the resources and the activities vary, and the graph is adapted to the change. The ontology-driven abstraction of activities is one of strong points in favor of our approach, since it leverages the abstraction to the business process.

Defining a process at a higher level of abstraction enables the recognition or the description of flows at a lower level of abstraction. This requires a compromise with reference to the soundness of the algorithm, since by starting from the event log we consider only a limited number of traces with respect to all possible ones related to the process. Fig. 3 shows the functional model of the application.

3.3. Activity Diagrams for process modeling

Activity Diagrams describe the sequencing of activities, with support for both conditional and parallel behavior. An activity diagram is a variant of a state diagram in which most, if not all, the states are activity states. Thus, much of the terminology follows that of state diagrams. Conditional behavior is delineated by branches and merges. A **branch** has a single incoming transition and several guarded outgoing transitions. Only one of the outgoing transitions can be taken, so the guards should be mutually exclusive. Using *[else]* as a guard indicates that the "else"transition should be used if all the other guards on the branch are false. A **merge** has multiple input transitions and a single output. A merge marks the end of conditional behavior started by a branch. It is not necessary to show the explicit diamond for branches and merges. An activity state, like any state, can have multiple guarded output transitions and multiple input

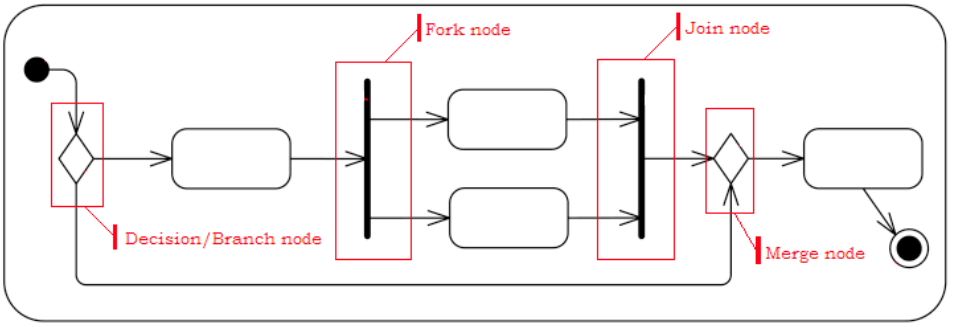

Fig. 4. Control nodes overview.

transitions. Diamonds need to be used just to make the branches and merges clear in the diagram. Parallel behavior is indicated by forks and joins. A **fork** has one incoming transition and several outgoing transitions. When the incoming transition is triggered, all of the outgoing transitions are taken in parallel.

In a parallel behavior, synchronization is needed this can be obtained with a **join** before the outgoing transition. With a join, the outgoing transition is taken only when all the states on the incoming transitions have completed their activities. Forks and joins must match. In the simplest case, this means that every time there is a fork, there must be also a join that joins together the threads started by that fork.

Fig. 4 shows the just described AD control nodes. Depending on the value of *AL* in the ontology we obtain a different Activity Diagram. Fig. 5 and Fig. 6 show the Activity Diagram produced for the example described previously. In Fig. 5 the AD refers to the high level of abstraction, i.e. by considering only the activities extracted from the event log file, whereas in Fig. 6 we show the AD of the process obtained by considering the resources performing the activities. The Activity diagram is traced using swimlanes. By using swimlanes, the activity diagrams is arranged into vertical zones separated by lines. This solution conveys which resource is responsible for each activity.

Let us consider as example for the use of resources, the cases 1 and 4:

Case 1

- `a: register request, Pete;`
- `b: examine thoroughly, Sue;`
- `d: check ticket, Mike;`
- `e: decide, Sara;`
- `f: reject request, Pete;`

Case 4

- `a: register request, Pete;`
- `d: check ticket, Mike;`
- `b: examine thoroughly, Sean;`
- `e: decide, Sara;`
- `f: reject request, Ellen.`

The obtained activity diagram has swimlane label with the names of the resources: `Pete, Sue, Mike, Sara, Sean` and `Ellen`.

The activity `register request` is performed by `Pete` in both the cases; the `examine thoroughly` is performed by `Sue` in the case 1 so the activity is in the corresponding swimlane and by `Sean` in case 4. Performer of activity `check ticket` is `Mike` in both the cases; the same is for `decide`. Finally, the `reject request` is executed by `Pete` and by `Ellen`; hence there are two occurrences of the same activity in two different swimlane. The use of AD operators enables the tracing of the two cases in the same activity diagram.

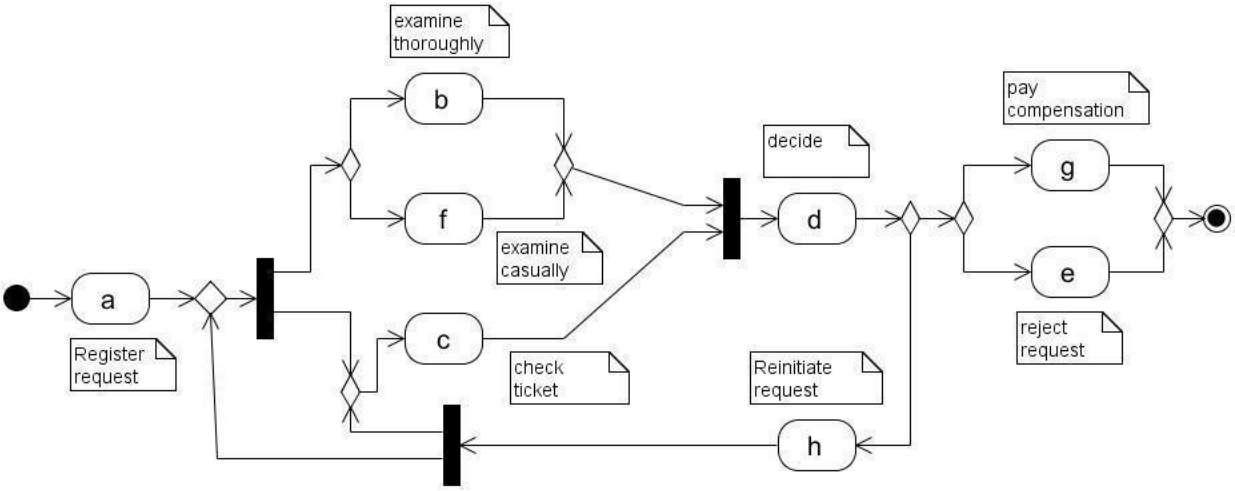

Fig. 5. Activity Diagram obtained at high level of abstraction.

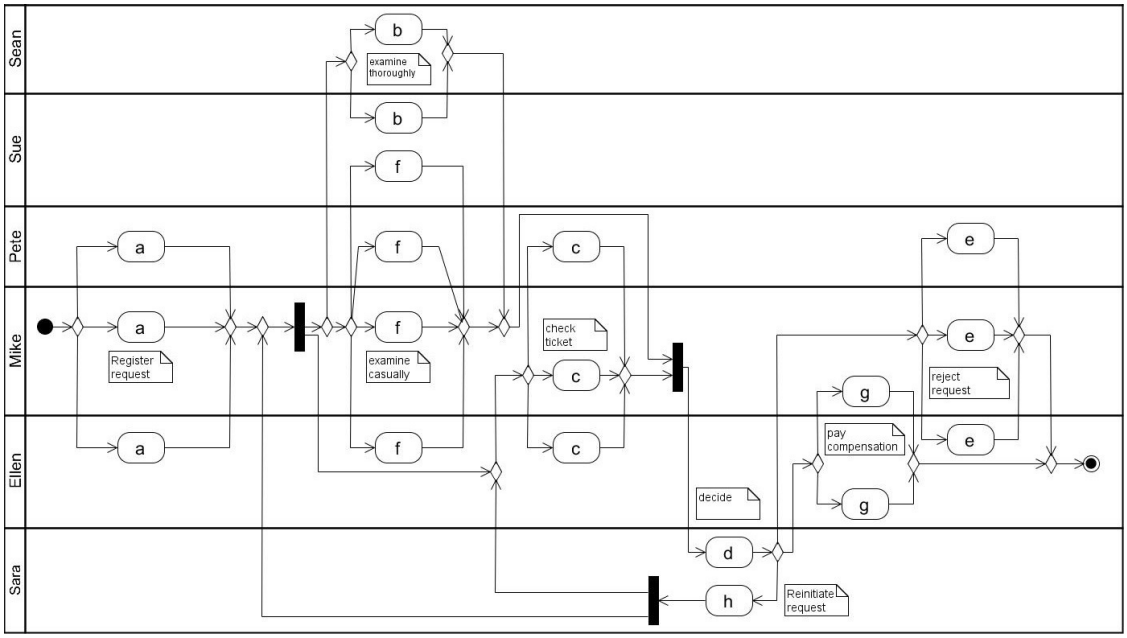

Fig. 6. Activity Diagram obtained at low level of abstraction.

4. Related Work

In the last years several academic and commercial tools have been developed for Business Activity Monitoring and Business Process Intelligence. Among the most famous we recall EMiT, Little Thumb, InWoLvE, Process Miner, and MinSoN on the academic side and ARIS PPM, HP BPI, and ILOG JViews on the commercial one. Very often, the main drawback of these tools is that they use different formats for reading/storing log files and present their results in different ways. An interesting and powerful system available in the literature is ProM[4]: the tool extracts data from event logs and produces a Petri Net as output to describe the process model. The ProM framework overcomes previously described limitations by using a flexible format with respect to the input and output. We use the same input

type – the event log – but we propose a more flexible output that is based on an XML format and a modified version of the alpha algorithm. With respect to the ProM framework[4], our approach analyze the event logs; in fact, it provide a flexible and customizable system depending on the output format to be used, on the resources to be considered and on the desired level of abstraction. In this stage of the work we use a properly defined subset of XMI (which results much more complex than we need) with a simplified syntax. With respect to existing tools for process mining we choose to produce an activity diagram for several reasons that we briefly explain in the following, that are both drawbacks of other approaches and advantages of AD. First of all, activity diagrams can handle parallel processes; this is important for business modeling since it encourages parallel behavior and removes unnecessary sequences in people's behavior. Doing things in parallel can improve the efficiency and responsiveness of business processes. Using AD it is possible to use XMI Metadata Interchange (XMI)[5] format for representing diagram and provide useful translation in other formalisms or languages.

A Document Type Definition (DTD) to represent UML AD in XMI format is also proposed[10]. The DTD considers the activity diagram as a graph. The elements in the activity diagram are nodes and edges. The proposed DTD is based on UML 2.0, where the activity diagram semantics is rooted in Petri Nets[11] rather than state machines, of the older version of UML. Each node and edge in the activity diagram can be mapped to the respective XMI tags using this DTD; the DTD is used in the conversion of activity diagram to XMI format.

Several state of the art approaches use an ontology to model knowledge about processes or events for process mining purposes. M. Hepp et al. use a knowledge base approach for modeling the Semantic Business Process domain[12]. The work of Hepp proposes a set of ontologies and formalisms, and defines the scope of these ontologies by giving competency questions, hence proposes an ontology engineering process, rather than a tool for process mining as we do. In fact, with respect to that work, we implement a tool that extracts knowledge from events and builds the process model starting from that information. In our approach, the ontology is the support to structure classes related to concept of the given domain and is integrated in the implemented tool. In the work by Caetano et al.[13], the authors show that combining process mining with enterprise ontology contributes to the analysis of business processes, especially in terms of determining the boundaries of authority and responsibility of the process. With respect to that proposal we use a wider approach since we implement a tool that can be further extended both to different domain ontology and to different modeling languages. The paper by Sellami et al.[14] proposes to semantically annotate event log based on a domain ontology. The ontology allowed to remove many ambiguities about data in event log and to share organizational knowledge. The approach was implemented within the ProM framework hence the format of the output is a Petri Net. With respect to this tool we use an AD as format for the output with the previously advantages described. The work by Nykanen et al.[15] mines event logs to construct an ontology that encodes log events as individuals associated with property values. With respect to that approach we use a more flexible output format, since the XMI obtained from the AD can be easily translated in different languages. The paper by Brisson L. and Collard M.[16] presents a methodology, named KEOPS, for integrating expert knowledge in the data mining. The approach is ontology-driven, anyway they propose an approach for data mining that extracts knowledge about data from stored data. With reference to that approach we perform process mining so our ontology-driven method extracts processes from events instead of knowledge from data.

Other approaches and languages for business and workflow modeling are XPDL[17], BPMN[18], ebXML BPSS[19]. XML Process Definition Language (XPDL)[17] is the language proposed by the Workflow Management Coalition (WfMC) to interchange process definitions between different workflow products. XPDL presents several drawbacks as highlighted by van der Aalst that analyzes XPDL using a set of 20 basic workflow patterns and expose some of the semantic problems. Business Process Modeling Notation (BPMN)[20] developed by Business Process Management Initiative (BPMI), since 2005 is maintained by the Object Management Group (OMG), after the merger between this organization and BPMI. In January 2011, OMG released BPMN version 2.0 which extends the scope and capabilities of the previous version, BPMN 1.2, in several areas: formalizes the execution semantics for all BPMN elements, defines an extensibility mechanism for both Process model extensions and graphical extensions, refines event composition and correlation, extends the definition of human interactions and defines a Choreography model[18]. In the work by Geambasu[21] an evaluation focused on the two most widely used graphical notations for business processes – BPMN and UML AD – is performed. The evaluation criteria are: capacity of being readily understandable, adequacy of the graphical elements of BPMN and UML AD to represent the real business processes of an organization and mapping to Business Process Execution Languages (BPEL). The results of evaluating BPMN and UML AD against

each of these three criteria show an almost similar behavior of the two formalisms. The paper[22] reviews how the two graphical process modeling notations can represent the workflow patterns. The solutions of the two notations are compared for technical ability to represent the patterns as well as their readability. The fact that both notations provide similar solutions to most of the patterns indicates how close the notations are in their presentation. They both share many of the same shapes for the same purposes (e.g., rounded rectangles for activities, diamonds for decisions, etc.). There are some differences in modeling object shapes, with BPMN containing fewer core objects and then having variations on these objects to handle the complexities that might arise in modeling processes, as shown with the above workflow patterns. Since the Activity Diagram and Business Process Diagram are very similar and are views for the same metamodel, it is possible that they will converge in the future. A complete comparison of AD and BPMN is performed[23]. The execution and simulation of BPMN models provide a number of similarities to UML AD, anyway for workflow modeling, Activity Diagram proves to be more suitable. The authors of the work[24] discuss the design choices that underlie the semantics of this two languages and investigate whether these design choices are met in low-level and high-level Petri net semantics. The main difference between the Petri net semantics and the semantics of UML ADs is that the Petri net semantics models closed and active systems that are non-reactive, whereas semantics of UML ADs models open and reactive systems. Since workflow systems are open and reactive systems, we conclude that Petri nets are not entirely suitable for workflow modelling. As result of the comparison among different languages for modelling workflow, we choose AD as output format for our tool because all the advantages emerged from this study.

5. Conclusions and Future Work

In this paper we describe PrOnto: a prototype system able to discover business processes from an event log and a suitable level of abstraction chosen in a related business ontology. The tool is an ontology-driven support for process monitoring and analysis. PrOnto reads activities for a process by mining the event log file of processes and extracts a model of the process described in a UML Activity Diagram model. The framework is flexible and adaptable to different languages for business models being based on the XML syntax of Activity Diagram. An ontology models resources and activities of the processes and allows different levels of abstraction in the modeling. Extraction of business model is performed by properly varying the *alpha algorithm* for process mining in order to encode the XML of the AD. We present a running example to validate the approach.
We are testing event log's mining aimed at measuring the two strengths of the tool: the Involvement of the user in managing the ontology with her feedback and the appropriateness of the ontology relative to the different levels of abstraction that can be considered.
Also, we are currently performing experiments and evaluation of the approach using the sets of logs provided by an Italian company, Corvallis S.P.A. for "Corvallis 3.0 " project. Moreover, we are developing a new version of PrOnto as an add-on component to integrate in the PROM framework. As far as the modeling is concerned, we are working to include in the ontology, a process control mined on security issues. We are also working to use different abstraction levels corresponding to the given number of relationships considered in the ontology. The sets of event logs provided by the company are being used to test the efficiency algorithm and the performance indexes and metrics of search and retrieval in the modeled ontology.

Acknowledgements

The authors acknowledges support of "Corvallis 3.0" project. The author Francesco Nocera acknowledges support of Exprivia S.p.A Ph.D grant 2016.

References

1. van der Aalst, W.M.. *Process mining: data science in action*. Springer; 2016.
2. Funk, M., Rozinat, A., Alves de Medeiros, A., van der Putten, P., Corporaal, H., van der Aalst, W.. Semantic concepts in product usage monitoring and analysis. *ES Reports, Sep* 2008;**29**:20.
3. Finin, T., Joshi, A., Kagal, L., Niu, J., Sandhu, R., Winsborough, W., et al. R owl bac: representing role based access control in owl. In: *Proceedings of the 13th ACM symposium on Access control models and technologies*. ACM; 2008, p. 73–82.

4. Van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H., Weijters, A., Van Der Aalst, W.M.. The prom framework: A new era in process mining tool support. In: *International Conference on Application and Theory of Petri Nets*. Springer; 2005, p. 444–454.
5. Group., O.M.. Xml metadata interchange (xmi). version 2.0.; May 2003.
6. Rockstrom, A., Saracco, R.. Sdl–ccitt specification and description language. *IEEE Transactions on Communications* 1982;**30**(6):1310–1318.
7. Van Der Aalst, W., Van Hee, K.M.. *Workflow management: models, methods, and systems*. MIT press; 2004.
8. Agrawal, R., Gunopulos, D., Leymann, F.. Mining process models from workflow logs. In: *International Conference on Extending Database Technology*. Springer; 1998, p. 467–483.
9. Van der Aalst, W., Weijters, T., Maruster, L.. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 2004;**16**(9):1128–1142.
10. Philip Samuel, S.E.. Document type definition for the xmi representation of uml2. 0 activity diagram. *International Journal of Recent Trends in Engineering* 2009;**1**(1).
11. Murata, T.. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 1989;**77**(4):541–580.
12. Hepp, M., Roman, D.. An ontology framework for semantic business process management. *Wirtschaftinformatik Proceedings 2007* 2007; :27.
13. Caetano, A., Pinto, P., Mendes, C., da Silva, M.M., Borbinha, J.. Analysis of business processes with enterprise ontology and process mining. In: *Enterprise Engineering Working Conference*. Springer; 2015, p. 82–95.
14. Sellami, R., Gaaloul, W., Moalla, S.. An ontology for workflow organizational model mining. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. IEEE; 2012, p. 199–204.
15. Nykänen, O., Rivero-Rodriguez, A., Pileggi, P., Ranta, P.A., Kailanto, M., Koro, J.. Associating event logs with ontologies for semantic process mining and analysis. In: *Proceedings of the 19th International Academic Mindtrek Conference*. ACM; 2015, p. 138–143.
16. Brisson, L., Collard, M.. An ontology driven data mining process. In: *International Conference on Enterprise Information Systems*. 2008, p. 54–61.
17. Interface, W.P.D.. Workflow management coalition workflow standard workflow process definition interface–xml process definition language; 2001.
18. Model, O.B.P.. Notation (bpmn). object management group, formal; 2011.
19. Clark, J., Casanave, C., Kanaskie, K., Harvey, B., Smith, N., Yunker, J., et al. ebxml business process specification schema version 1.01. Tech. Rep.; Technical report, UN/CEFACT and OASIS; 2001.
20. White, S.A.. Introduction to bpmn. *IBM Cooperation* 2004;**2**(0):0.
21. Geambasu, C.V.. Bpmn vs. uml activity diagram for business process modeling. *Accounting and Management Information Systems* 2012; **11**(4):637.
22. White, S.A.. Process modeling notations and workflow patterns. *Workflow handbook* 2004;**2004**:265–294.
23. Peixoto, D., Batista, V., Atayde, A., Borges, E., Resende, R., Pádua, C.. A comparison of bpmn and uml 2.0 activity diagrams. In: *VII Simposio Brasileiro de Qualidade de Software*; vol. 56. 2008, .
24. Eshuis, R., Wieringa, R.. Comparing petri net and activity diagram variants for workflow modelling–a quest for reactive petri nets. In: *Petri Net Technology for Communication-Based Systems*. Springer; 2003, p. 321–351.