Declarative Process Mining with the Declare Component of ProM

Fabrizio Maria Maggi[⋆]

University of Tartu, Estonia
f.m.maggi@ut.ee

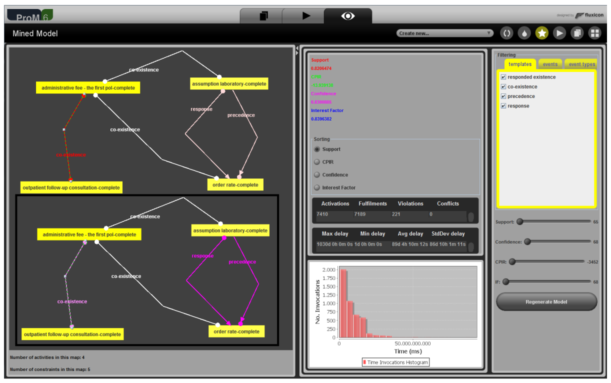
Abstract. Traditional process mining techniques mainly work with procedural modeling languages (like Petri nets) where all possible orderings of events must be specified explicitly. However, using procedural process models for describing processes working in turbulent environments and characterized by a lot of variability (like healthcare processes) is extremely difficult. Indeed, these processes involve several possible execution paths and representing all of them explicitly makes the process models quickly unreadable. Using declarative process models (like Declare) ensures flexibility in the process description. Even processes that work in environments where participants have more autonomy and are, therefore, more unpredictable can be represented as compact sets of rules. In the process mining tool ProM, there are several plug-ins that can be used for different types of analysis based on Declare ranging from the discovery of Declare models, to conformance checking, to the online monitoring of running process instances.

1 Introduction

Process discovery, conformance checking and process model enhancement are the three basic forms of process mining [9]. In particular, process discovery aims at producing a process model based on example executions in an event log and without using any a-priori information. Conformance checking pertains to the analysis of the process behavior as recorded in an event log with respect to an existing reference model. Process model enhancement aims at enriching and extending existing process models with information retrieved from logs, e.g., a process model can be extended with performance-related information such as flow time and waiting time.

In the last years, several works have been focused on process mining techniques based on declarative process models [2–4, 6–8]. These techniques are very suitable for processes characterized by high complexity and variability due to the turbulence and the changeability of their execution environments. The dichotomy procedural versus declarative can be seen as a guideline when choosing

[⋆] The author would like to thank R.P. Jagadeesh Chandra Bose, Andrea Burattin, Massimiliano de Leoni and Luciano García-Bañuelos for their collaboration in the implementation of the plug-ins presented in this paper.


ProM 6
designed by fluxicon
Mined Model
Create new...
co-existence
administrative fee - the first pol-complete
assumption laboratory-complete
response
precedence
outpatient follow-up consultation-complete
order rate-complete
No. Invocations
Time (ms)
Time Invocations Histogram
Regenerate Model

Support
0.8206474
CPIR
-13.828130
Confidence
0.8300885
Interest Factor
0.8396382

Sorting
[✓ Support]
CPIR
Confidence
Interest Factor

Activations	Fulfilments	Violations	Conflicts
7410	7189	221	0

Max delay	Min delay	Avg delay	StdDev delay
1030d 0h 0m 0s	1d 0h 0m 0s	89d 4h 10m 12s	86d 10h 1m 11s

Filtering: templates | events | event types
- ✓ responded existence
- ✓ co-existence
- ✓ precedence
- ✓ response

Support: 65
Confidence: 68
CPIR: -3452
IF: 68

Number of activities in this map: 4
Number of constraints in this map: 5

Fig. 1. Screenshot from the Declare Maps Miner.

the most suitable language to represent models in process mining algorithms: process mining techniques based on procedural languages can be used for predictable processes working in stable environments, whereas techniques based on declarative languages can be used for unpredictable, variable processes working in turbulent environments.

The plug-ins presented in this paper use the *Declare* notation [10]. Declare is a declarative language that combines a formal semantics grounded in Linear Temporal Logic (LTL) on finite traces. A Declare model consists of a set of constraints which, in turn, are based on templates. Templates are abstract entities that define parameterized classes of properties and constraints are their concrete instantiations.

In this paper, we illustrate how the presented plug-ins work by showing their application in the context of the BPI Challenge 2011 [1] that records the treatment of patients diagnosed with cancer from a large Dutch hospital. This demonstrates that they are viable to handle event logs of real-life size.

2 Plug-ins Description

Through the *Declare Maps Miner*, it is possible to generate from scratch a set of Declare constraints representing the actual behavior of a process as recorded in an event log. The user selects from a list of Declare templates the ones to be used for the discovery task, i.e., the mined model will contain only constraints that are instantiations of the selected templates. In addition, the user can provide the plug-in with some a-priori knowledge (if it is available), to guide the discovery task. For example, the user can decide to limit the possible constraints to discover. This allows her to generate only constraints that are the most interesting from the domain point of view, thus reducing the complexity of the resulting models.

There are two mechanisms for limiting the possible constraints to discover. First, it is possible to ignore constraints between event types (such as *start* and *complete*) of the same activity.[1] Secondly, the user can provide different groups of activities (each group can be loaded in the form of a list of activities in a text file) and specify if only intra-group or inter-group constraints should be considered. Intra-group constraints refer to the class of constraints where the activities involved all emanate from a single group. In many scenarios, analysts would be interested in finding constraints between activities pertaining to a functionality, to a particular department in an organization, etc. For example, in a hospital event log, an analyst would be interested in finding relationships/constraints between the various administration activities. Inter-group constraints refer to the class of constraints where the activities involved belong to two different groups. For example, in a hospital log, an analyst would be interested in constraints between activities involved in surgery and therapy.

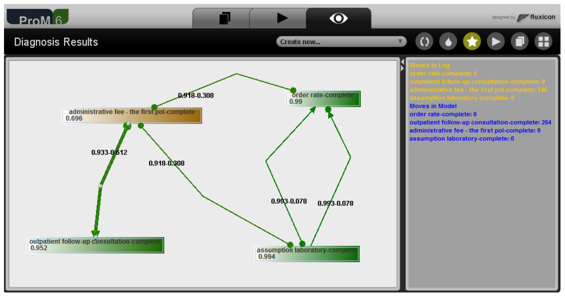
The user can also specify thresholds for parameters *minimum support* and *alpha*. Parameter minimum support allows her to select the percentage of traces in which a constraint must be satisfied to be discovered (and to filter out noisy traces). Parameter alpha can be used ignore constraints that are trivially true in the discovery task. For example, constraint *response(A,B)*, specifying that if *A occurs*, then eventually *B follows*, is trivially true in process instances in which *A* does not occur at all.

Fig. 1[2] shows the output produced by the Declare Maps Miner. The discovery results are presented to the user as interactive maps and give different information about the process. Activities (shown as rectangles) are colored based on their frequency (from white indicating low frequency to yellow indicating high frequency). The user can highlight, in the discovered maps, the Declare constraints (shown as arcs between activities) that are more relevant (based on different metrics) or prune out the ones that are less interesting, redundant or deriving from noise in the log. The maps produced by the Declare Maps Miner also show delays, latencies, time distances between activities and several statistics related to the temporal dimension of the process.

If an existing Declare model is already available, it is possible to repair it based on the information retrieved from a log (plug-in: *Repair a Declare Model*). The user can decide which aspects must be kept in the original set of rules and which ones can be discarded. For example, the user can decide to keep only certain types of constraints or constraints involving certain activities. It is possible to just remove less interesting or redundant rules or to strengthen (if possible) the constraints included in the original set. Starting from a Declare model and from a log, the Declare model can be extended with time information (plug-in: *Extend a Declare Model with Time Information*). Through the *Data-Aware Declare Miner*, it is also possible to discover constraints enriched with data conditions.

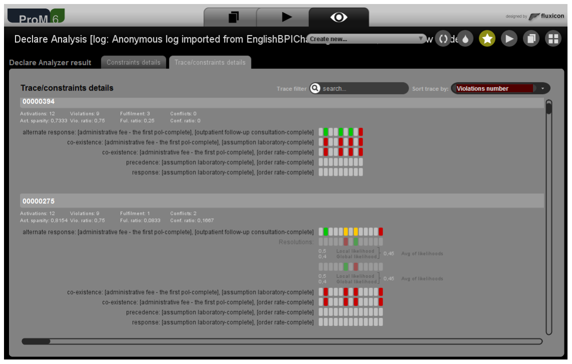
[1] The miner is able to deal with non-atomic activities and to discover constraints involving different parts of the activities' lifecycle.

[2] For space reasons, the quality of the screenshots here presented is not the best. However, all the figures included in this paper are available at `http://math.ut.ee/~fabrizio/BPMdemo13/demo/FiguresDemo2013.zip`



Moves in Log
order rate-complete: 0
outpatient follow-up consultation-complete: 0
administrative fee - the first pol-complete: 746
assumption laboratory-complete: 0
Moves in Model
order rate-complete: 0
outpatient follow-up consultation-complete: 264
administrative fee - the first pol-complete: 0
assumption laboratory-complete: 0

Fig. 2. Screenshot from the Declare Diagnoser.



Declare Analysis [log: Anonymous log imported from EnglishBPICh

Trace/constraints details

00000394

```
Activations: 12          Violations: 9       Fulfilment: 3     Conflicts: 0
Act. sparsity: 0,7333    Vio. ratio: 0,75    Ful. ratio: 0,25  Conf. ratio: 0
```

alternate response: [administrative fee - the first pol-complete], [outpatient follow-up consultation-complete]
co-existence: [administrative fee - the first pol-complete], [assumption laboratory-complete]
co-existence: [administrative fee - the first pol-complete], [order rate-complete]
precedence: [assumption laboratory-complete], [order rate-complete]
response: [assumption laboratory-complete], [order rate-complete]

00000275

```
Activations: 12          Violations: 9       Fulfilment: 1       Conflicts: 2
Act. sparsity: 0,8154    Vio. ratio: 0,75    Ful. ratio: 0,0833  Conf. ratio: 0,1667
```

alternate response: [administrative fee - the first pol-complete], [outpatient follow-up consultation-complete]
Resolutions:
0,5 Local likelihood
0,4 Global likelihood
0,45 Avg of likelihoods
0,5 Local likelihood
0,4 Global likelihood
0,45 Avg of likelihoods
co-existence: [administrative fee - the first pol-complete], [assumption laboratory-complete]
co-existence: [administrative fee - the first pol-complete], [order rate-complete]
precedence: [assumption laboratory-complete], [order rate-complete]
response: [assumption laboratory-complete], [order rate-complete]

Fig. 3. Screenshot from the Declare Analyzer.

The *Declare Replayer* and the *Declare Diagnoser* (*Declare Checker* package) can be used to check the conformance of the actual behavior of a process as recorded in an event log with respect to a reference Declare model. The Declare Replayer generates a set of alignments between the log and the Declare model, i.e., information about what must be changed in the log traces to make them perfectly compliant with the model. The Declare Diagnoser projects the results obtained through the Declare Replayer onto the reference model (as shown in Fig. 2). This projection produces a map in which the critical activities/constraints of the reference model are highlighted. In particular, activities are colored from red to green according to the number of moves they are involved in, i.e., according to how many times they were in the wrong place in the log or missing when required. Constraints are colored from red to green based on the number of moves that are needed to make the log compliant to them. Through the *Declare Analyzer* (Fig. 3), the user can pinpoint where the process execution deviates from the reference Declare model and quantify the degree of conformance of the process behavior through several metrics, e.g., *fulfillment ratio* and *violation ratio*.

For completeness, we mention that, with *Mobucon LTL* (a provider of the operational support service in ProM), it is also possible to monitor a running process with respect to a reference Declare model and get information about its compliance at runtime. However, this functionality has already been presented in [11].

3 Summary

The plug-ins belonging to the Declare component of ProM are advanced prototypes and, as demonstrated through their application to the log provided for the BPI challenge 2011, are able to handle logs of real-life size. The Declare Maps Miner and Mobucon LTL have been used in the Poseidon project (`http://www.esi.nl/short/poseidon/`) to monitor sea vessels. In particular, in [5], we show how it is possible to construct Declare models from real-life historical data and monitor live data with respect to the mined model.

We consider the set of ProM plug-ins presented here stable enough for evaluating the declarative approach in real-life case studies. Binaries and source code can be obtained from the ProM repository (`https://svn.win.tue.nl/repos/prom`) and a release of each plug-in including the functionalities described in this paper is included in the nightly build version of ProM (`www.processmining.org`). A screencast of this demo can be found at `http://math.ut.ee/~fabrizio/BPMdemo13/demo/demo.html`. A flyer with a summary of the functionalities described in this paper is available at `http://math.ut.ee/~fabrizio/BPMdemo13/demo/DeclareFlyer.pdf`.

References

1. 3TU Data Center. BPI Challenge 2011 Event Log, 2011. doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54.
2. A. Burattin, F.M. Maggi, W.M.P. van der Aalst, and A. Sperduti. Techniques for a Posteriori Analysis of Declarative Processes. In *EDOC 2012*.
3. Massimiliano Leoni, Fabrizio Maria Maggi, and WilM.P. Aalst. Aligning event logs and declarative process models for conformance checking. In *BPM 2012*.
4. F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst. Runtime verification of LTL-based declarative process models. In *RV 2011*.
5. Fabrizio M. Maggi, Arjan J. Mooij, and Wil M. P. van der Aalst. *Analyzing Vessel Behavior using Process Mining*, chapter Poseidon book.
6. F.M. Maggi, J.C. Bose, and W.M.P. van der Aalst. Efficient discovery of understandable declarative models from event logs. In *CAiSE 2012*.
7. F.M. Maggi, R.P.J.C. Bose, and W.M.P. van der Aalst. A knowledge-based integrated approach for discovering and repairing declare maps. In *CAiSE 2013*.
8. F.M. Maggi, M. Montali, M. Westergaard, and W.M.P. van der Aalst. Monitoring business constraints with linear temporal logic: An approach based on colored automata. In *BPM 2011*.
9. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
10. W.M.P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - R&D*.
11. Michael Westergaard and Fabrizio Maria Maggi. Declare: A tool suite for declarative workflow modeling and enactment. In *BPM (Demos)*, 2011.