# Rule Mining in Action: The RuM Toolkit

Anti Alman
University of Tartu
Estonia
anti.alman@ut.ee

Claudio Di Ciccio
Sapienza University of Rome
Italy
diciccio@di.uniroma1.it

Dominik Haas
WU Vienna
Austria
dominik.haas@s.wu.ac.at

Fabrizio Maria Maggi
Free University of Bolzano
Italy
maggi@inf.unibz.it

Jan Mendling
WU Vienna
Austria
jan.mendling@wu.ac.at

***Abstract*—Procedural process modeling languages can be difficult to use for process mining in cases where the process recorded in the event log is unpredictable and has a high number of different branches and exceptions. In these cases, declarative process modeling languages such as DECLARE are more suitable. Declarative languages do not aim at modeling the end-to-end process step by step, but constrain the behavior of the process using rules thus allowing for more variability in the process model yet keeping it compact. Although there are several commercial and academic process mining tools available based on procedural models, there are currently no comparable tools for working with declarative models. In this paper, we present RuM, an accessible and easy-to-use rule mining toolkit integrating multiple DECLARE-based process mining methods into a single unified application. RuM implements process mining techniques based on Multi-Perspective DECLARE, namely the extension of DECLARE supporting data constraints together with control-flow constraints. In particular, RuM includes support for process discovery, conformance checking, log generation and monitoring as well as a model editor. The application has been evaluated by conducting a qualitative user evaluation with eight process analysts.**

***Index Terms*—Rule Mining, Process Analytics Tool, Declarative Process Models, Process Mining**



## I. INTRODUCTION

Process mining is the area of Business Process Management (BPM) which is focused on the analysis of business processes based on event logs containing information about process executions. A key artifact used in process mining is a process model. The most common type of process models used for process mining are procedural models, which aim at describing end-to-end processes and allow only for activities that are explicitly triggered through the control-flow. However, modeling the entire control-flow step by step can be inconvenient in some cases. For example, if the process is unpredictable and has a high number of different branches and exceptions, the models could become quickly unreadable. In these cases, it may be better to use declarative process models that model the process as a set of rules that the process should follow. Although there are several commercial and academic process mining tools available based on procedural models, there are currently no comparable tools for working with declarative models.

In this paper, we present RuM, a novel and easy to use process mining toolkit based on the declarative modeling language DECLARE [1]. RuM implements process mining techniques for both standard DECLARE and MP-DECLARE [2], the latter being the multi-perspective extension of DECLARE supporting data constraints together with control-flow constraints. RuM also provides a model editor that is fully MP-DECLARE compliant and equipped with a chatbot that supports inexpert users in defining DECLARE constraints using natural language expressions.

To assess the feasibility of RuM, we conducted a qualitative user evaluation of which we sketch here the main findings.[1] In general, the application was well received and it was recognized to be timely and highly needed by all participants.

To download RuM and get access to additional information about the tool, please refer to the RuM project website: https://rulemining.org. A tutorial on the use of the tool is available at https://git.io/JUvw4. A short video screencast showing a quick overview of RuM is available at https://youtu.be/nXFNDDbOcU0. In [3], we describe the techniques implemented in RuM in detail. In addition to what presented in [3], we introduce in this paper a new functionality (process monitoring) and an additional tool that eases the management and retrieval of models and event logs (the inventory).

The remainder of this paper is structured as follows. Section II gives a short overview of the functionalities of RuM and lists the process mining techniques available in the application. Section III describes the maturity of the tool by summarizing the results of our user evaluation and concludes the paper by spelling out directions for future work.

## II. FUNCTIONAL OVERVIEW

RuM is the first software platform natively designed to analyze declarative, rule-based process models. To that end, we have integrated and improved existing prototypes, but also created completely new features that enhance the user experience during the process analysis. To cater for the interoperability of the tool, we resort on existing standards for input and output files, namely XES [4] for the event logs and *decl* [5] for the models.

Figure 1 shows the home screen of the tool, giving direct access to its main functionalities. In the following, we describe those functionalities and the inventory, with which the user can save, restore and alter the diverse artifacts imported from the file system or created during the mining and analysis of processes and event logs. For the

[1] For a full description of the user evaluation the reader is referred to [3].

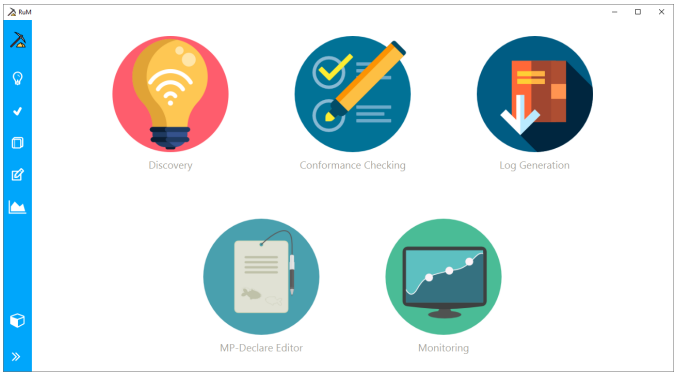


Fig. 1. The home screen of RuM with access to the discovery (Section II-A), conformance checking (Section II-B), log generation (Section II-C), MP-DECLARE editor (Section II-D), and monitoring (Section II-E) functionalities. The button to open the inventory is depicted as a box in the bottom-left corner (Section II-F).

sake of space, we shall delve more in the detail of the novel monitoring functionality and the inventory. For further information on the whole set of techniques, please refer to [3], the online tutorial (https://git.io/JUvw4) and video (https://youtu.be/nXFNDDbOcU0).

### A. Discovery

Figure 2 illustrates the user interface (UI) of the discovery panel applied to an example event log describing the treatment of patients suffering the from the sepsis disease in a hospital. Four methods are available for process discovery: Declare Miner [6], MINERful [7], MP-Declare Miner [8] and MP-MINERful. The MP variants add to the base mining algorithms a post processing step for discovering data conditions.[2] We remark that MP-MINERful is a novel integration of MINERful with the post processing step, created specifically for RuM.

The discovery results can be explored by using three complementary views: through a process map (DECLARE view), a textual description (textual view), or as a procedural model (automaton view).[3] We remark that these views support filtering based on activity support and constraint support thus providing the possibility for users to show/hide outlier behaviors. The filters are applied on the fly without rediscovering the model to speed up the user interaction.

### B. Conformance Checking

RuM implements three techniques for conformance checking: Declare Analyzer [2], Declare Replayer [10] and Data-Aware Declare Replayer.[4] The Declare Analyzer takes as input a model and an event log, and returns constraint activations, violations, and fulfilments in the input log. The Declare Replayer and the Data-Aware Declare Replayer report trace alignments. The Data-Aware Declare Replayer can also account for the data perspective.

[2]Notice that all the "data-aware" versions of the techniques provided in RuM support a richer language at the expense of lower efficiency.

[3]The rendering of the automaton is based on the MINERful library [9].

[4]https://github.com/Clyvv/DataAwareDeclareReplayer

The conformance checking results are presented in groups. Each group displays the results for a specific trace or a specific constraint. For each group, the name of the group is displayed along with general descriptive statistics about the group. To foster interaction in the process analysis, the user can freely switch among groups and toggle an extended view for each group to see more details. This allows users to explore the results at a high level of detail while also being relatively compact in terms of user interface [3, Fig. 4].

### C. Log Generation

There are two log generation methods available in RuM: Alloy Log Generator [5] and MINERfulLog Generator [11]. The main difference between these methods from the user's standpoint is that the Alloy Log Generator can also account for the definition of data conditions in the input process model. Among other options, the user can specify the percentage of traces that trivially satisfy the constraints in the output log (that is, traces that comply with the constraints in the input model because their activations never occur) and the percentage of negative traces (i.e., traces that violate at least one of the constraints in the input model) [3, Fig. 6]. The generated log can be exported in the XES format.

### D. MP-DECLARE Editor

In order to provide a comprehensive toolset for working with DECLARE process models, RuM contains a model editor, which is the first one supporting both standard DECLARE and MP-DECLARE [3, Fig. 5]. The MP-DECLARE editor uses the *decl* file format for importing and exporting the models. All aspects of the format are supported: activity definitions, attribute definitions, activity-attribute bindings and constraints with all the allowed data and time conditions. The used visualization devices are the same as those of the model discovery panel, i.e., the model can be visualized using the standard DECLARE graphical notation, as text or in the form of an automaton. The editor is also equipped with a chatbot (Declo [12]) supporting the user in designing MP-DECLARE models using natural language.

### E. Monitoring

There are two monitoring methods available in RuM: MP-Declare with Alloy[5] and MobuconLTL [13]. The main difference between these methods from the user's standpoint is that MP-Declare with Alloy can also account for the definition of data conditions in the input process model.

The monitoring tool analyzes the state of each constraint and updates it for every event in a trace. The states follow the five-truth-values introduced for DECLARE in [14] and are visually depicted with different colors: temporarily satisfied (though currently satisfied, the constraint can become violated in the remainder of the trace: green), temporarily violated (vice-versa, it can become satisfied in the future: yellow), permanently satisfied (azure) or permanently violated (red). Orange indicates that there is a conflict between some of the

[5]https://github.com/b26140/Rule-mining-tool-with-monitor-extension

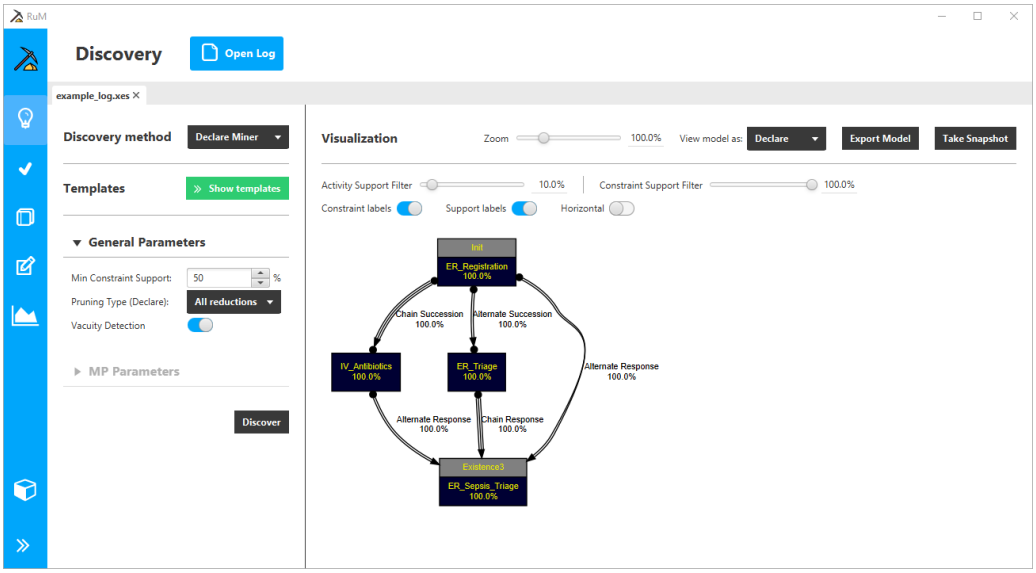


Fig. 2. Discovery UI with the DECLARE view.

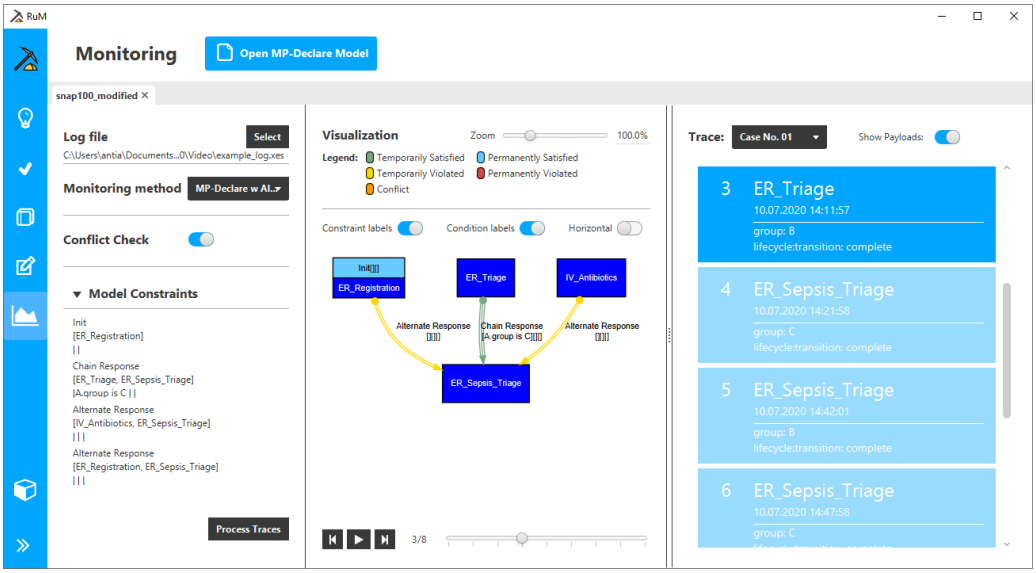


Fig. 3. Monitoring UI.

constraints, i.e., there is no possible sequence of future events that could satisfy all the conflicting constraints. The user can replay the whole trace, manually process the events in the trace one by one or jump to a specific event in the trace.

### F. The Inventory

Figure 4 illustrates the inventory, a novel artifact aimed at easing the management of the event logs and process models in use during the process analysis with RuM. All the files imported from the file system are retrievable from the inventory as well as all the discovered and edited process models and the generated logs. The users can save the intermediate results of their analysis as so-called *snapshots* so as to fetch them later on. Next to each snapshot, the inventory offers action buttons that can activate a related functionality: e.g., event logs can be directly re-routed as an input for discovery, whereas process models can be sent to the conformance checking, log generation, editor, or monitoring panels.

## III. MATURITY AND FUTURE REMARKS

A qualitative user study, presented in [3], was conducted to assess the feasibility of RuM. Our aim was to (1) gain insights into how users from different backgrounds perceived the application and (2) identify means for improving it. We selected eight process analysts as participants for our study. Four of the participants had little to no knowledge of DECLARE, the other four identified themselves as DECLARE experts. The study concluded with a post-survey questionnaire consisting of

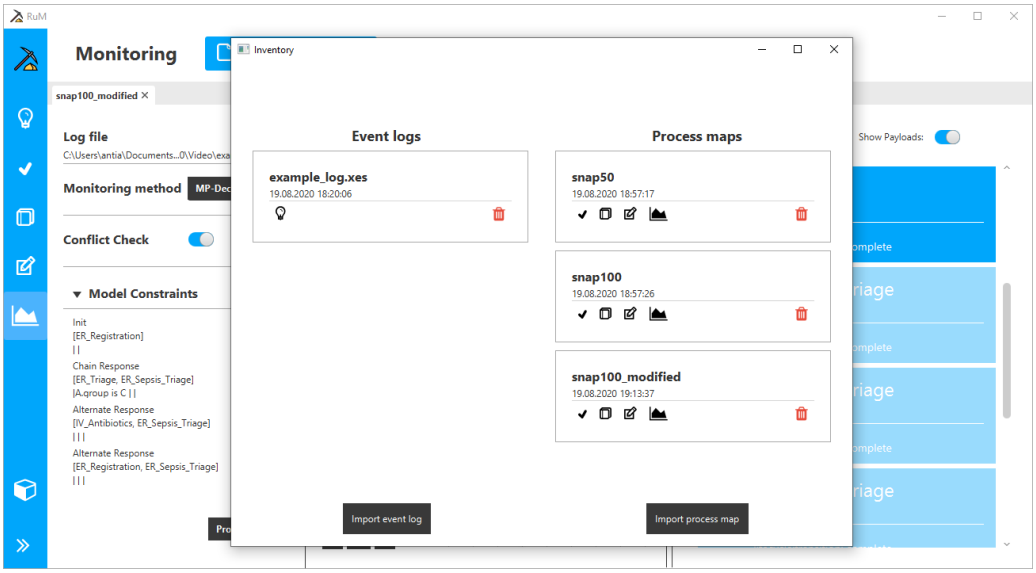


Fig. 4. The inventory.

multi-point Likert scales including the System Usability Scale (SUS) [15] and scales covering satisfaction, expectation confirmation, continuation intention, and usefulness. The average SUS score of 81.875 resulting from the evaluation of RuM shows that the toolkit was very well received and ready to be used in real scenarios with users having different backgrounds (a SUS score of 69.69 is considered average while a score above 80 is considered to be good or excellent [16]).

For future work, we aim at extending the user analysis upon the integration of the user feedback into our implementation of RuM. Furthermore, we will extend the current capabilities both from the perspective of the available functionalities (e.g., with the mining of branched-DECLARE constraints [17]) and visualization schemes (e.g., with the graphical editing of models).

## ACKNOWLEDGMENTS

The authors would like to thank all the participants of the user evaluation for taking the time to evaluate RuM and for providing us with invaluable feedback on how RuM can be improved in the future.

The work of A. Alman was partly supported by the Estonian Research Council (project PRG887). The work of C. Di Ciccio was partly supported by MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza University of Rome.

## REFERENCES

[1] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "DECLARE: Full support for loosely-structured processes," in *EDOC*, 2007, pp. 287–300.

[2] A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert systems with applications*, vol. 65, pp. 194–211, 2016.

[3] A. Alman, C. Di Ciccio, D. Haas, F. M. Maggi, and A. Nolte, "Rule mining with RuM," in *ICPM*, B. F. van Dongen, M. Montali, and M. T. Wynn, Eds. IEEE, 2020.

[4] C. W. Gunther and H. M. W. Verbeek, *XES - standard definition*, ser. BPM reports. BPMcenter.org, 2014, vol. 1409.

[5] V. Skydanienko, C. Di Francescomarino, C. Ghidini, and F. M. Maggi, "A tool for generating event logs from multi-perspective Declare models," in *BPM (Dissertation/Demos/Industry)*, 2018, pp. 111–115.

[6] F. M. Maggi, C. Di Ciccio, C. Di Francescomarino, and T. Kala, "Parallel algorithms for the automated discovery of declarative process models," *Inf. Syst.*, vol. 74, pp. 136–152, 2018.

[7] C. Di Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM Trans. Manage. Inf. Syst.*, vol. 5, no. 4, Jan. 2015.

[8] V. Leno, M. Dumas, F. M. Maggi, M. La Rosa, and A. Polyvyanyy, "Automated discovery of declarative process models with correlated data conditions," *Inf. Syst.*, vol. 89, p. 101482, 2020.

[9] J. Prescher, C. Di Ciccio, and J. Mendling, "From declarative processes to imperative models," in *SIMPDA*. CEUR-WS.org, Nov. 2014, pp. 162–173.

[10] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Sys.*, vol. 47, pp. 258–277, 2015.

[11] C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of Declare models," in *EOMAS*, 2015, pp. 20–36.

[12] A. Alman, K. J. Balder, F. M. Maggi, and H. van der Aa, "Declo: A chatbot for user-friendly specification of declarative process models," in *BPM Demos*, 2020.

[13] 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. Proceedings*, 2011, pp. 132–147.

[14] F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst, "Runtime verification of ltl-based declarative process models," in *Runtime Verification, 2011, Revised Selected Papers*, 2011, pp. 131–146.

[15] J. Brooke, *SUS-A quick and dirty usability scale. (in Usability Evaluation in Industry, by P.W. Jordan, B. Thomas, I.L. McClelland, B. Weerdmeester)*. CRC Press, June 1996, iSBN: 9780748404605.

[16] P. T. Kortum and A. Bangor, "Usability ratings for everyday products measured with the system usability scale," *International Journal of Human–Computer Interaction*, vol. 29, no. 2, pp. 67–76, 2013.

[17] C. Di Ciccio, F. M. Maggi, and J. Mendling, "Efficient discovery of Target-Branched Declare constraints," *Inf. Syst.*, vol. 56, pp. 258–283, 2016.