Rule Mining with RuM (Extended Abstract)

Anti Alman[1], Claudio Di Ciccio[2], and Fabrizio Maria Maggi[3]

[1] University of Tartu, Tartu, Estonia
anti.alman@ut.ee
[2] Sapienza University of Rome, Rome, Italy
claudio.diciccio@uniroma1.it
[3] Free University of Bozen-Bolzano, Bolzano, Italy
maggi@inf.unibz.it

Abstract. Declarative process modeling languages are especially suitable to model loosely-structured, flexible business processes. One of the most prominent of these languages is DECLARE. The DECLARE language can be used for all process mining branches and a plethora of techniques have been implemented to support process mining with DECLARE. The process mining application RuM integrates multiple DECLARE-based process mining methods into a single application and is developed to be the starting point for the use of DECLARE both in industry and academia. RuM has been evaluated by conducting a qualitative user evaluation, the results of which have been used as input for further development. In this paper, we give a short overview of the current functionalities of RuM, including the main improvements made thus far.

Keywords: Process analytics · Declarative modeling · Process mining · Process discovery · Conformance checking

1 Introduction

One of the obstacles for the adoption of declarative process mining techniques, especially in the context of the DECLARE language [23], has been a lack of comprehensive, easy-to-use process mining toolkits [22, RC7]. However, this shortcoming has been recently addressed with the introduction of RuM [3,2], which implements various process mining techniques based on DECLARE and its multi-perspective extension MP-DECLARE [6]. RuM is specifically developed to be the starting point for the use of DECLARE both in industry and academia.

To provide an intuition about the nature and expressiveness of DECLARE models, consider the example given in Fig. 1. This model is inspired by the analysis of a real-world event log of the process of handling patients affected by sepsis [18]. The process begins with the emergency room (ER) registration. After that, if the condition on the Systemic Inflammatory Response Syndrome (SIRS) criteria attribute holds true, then intravenous (IV) antibiotics have to be administered. The administration of IV antibiotics must follow the ER triage. The ER sepsis triage has to be executed immediately after ER triage, but with the condition that the actors carrying out the two activities differ (condition on `org:group`). Finally, The analysis of the presence of lactic acid requires that ER sepsis triage has occurred at most 3 hours before.

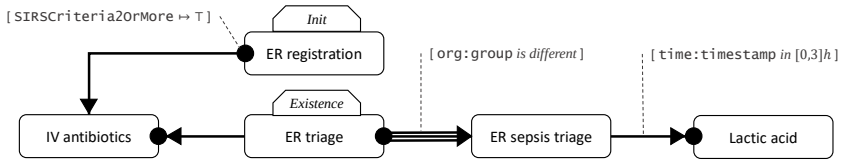

Fig. 1. An MP-DECLARE map

In this paper, we provide a short overview of the functionalities available in RuM to operate with process models similar to the one described above (including additional functionalities not covered in [3,2]). RuM is open-source and publicly available at https://rulemining.org.

2 Functional Overview

RuM integrates and improves multiple existing prototypes, while also providing novel features that can be found in RuM only. Interoperability of all modules is ensured by relying on existing standards, namely XES [12] for the event logs and *decl* [21] for the models. The main functionalities of RuM are: process discovery, conformance checking, log generation, model editor, and monitoring. Furthermore, RuM features an inventory system allowing for process models and event logs to be reused easily across the application. In the following, we describe the functionalities of RuM (including the inventory) in more detail. We remark that the conformance results and monitoring views, the model editor, and the inventory panel were designed and developed from ground-up specifically for RuM.

Discovery. Process discovery is implemented using Declare Miner [15] and MINERful [10], both of which can be further augmented by discovering data conditions (based on [14]) and time conditions (developed specifically for RuM). As depicted in Fig. 2(a), the discovered model can be explored by using three complementary views:[4]

DECLARE view: it employs the standard graphical notation of DECLARE. Activities are represented as rectangles, while constraints are represented either above the activities (unary constraints) or as lines between the activities (both with support percentages).

Textual view: it describes the model using natural language sentences that are easy to understand without any prior knowledge of DECLARE. This representation was developed specifically for RuM and meant for users who are less familiar with the standard graphical notation.

Automaton view: it displays the discovered process model as a finite-state automaton based on the a conjunction of the LTL_f formulas of each discovered constraint.

[4] All three views enable filtering based on activity support and constraint support.

(a) Process discovery (b) Conformance checking

(c) MP-Declare editor (d) Log generation

(e) Monitoring (f) Inventory

Fig. 2. Main functionalities of RuM

Conformance checking. The conformance checking functionality is based on three techniques: Declare Analyzer [6] detects activations, violations, and fulfillments; Declare Replayer [8] and Data-Aware Declare Replayer[5] report on trace alignments. As shown in Fig. 2(b), the results are presented in groups, each displaying the outcome for a specific trace or a specific constraint. For every group, its name is displayed along with general descriptive statistics. The user can freely switch among groups and toggle an extended view to read more details. In this way, users can explore the results at a high level of detail while also keeping the user interface relatively compact.

MP-Declare editing. RuM features the first fully MP-DECLARE compliant model editor, depicted in Fig. 2(c). Models are imported and exported in the *decl* file for-

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

mat [21], which allows for activity definitions, attribute definitions, activity-attribute bindings, DECLARE constraints, and data and time conditions of MP-DECLARE. The visualization of the entire model is updated on-the-fly and the same views (with minor modifications) are provided as in Section 2. Furthermore, the user can add constraints and data conditions using natural language sentences, which can be provided both via voice and text through a simple chatbot [1].

Log generation. To generate logs, RuM resorts to the AlloyLogGenerator [21] and MINERful Log Generator [7], the former of which can also account for the data conditions in the input process model. As illustrated in Fig. 2(d), the user can specify the percentage of vacuous traces (i.e., traces fulfilling the constraints because the constraints are never activated), the number of negative traces (i.e., traces violating at least one constraint), and the case attributes of the generated event log.

Monitoring. Figure 2(e) illustrates the monitoring panel, through which the user can analyze the state of each constraint for every event sequentially. The states follow the four-truth-values introduced in [17]: temporarily satisfied (satisfied given the events thus far, but can be violated in the future: green), temporarily violated (violated, but can be satisfied in the future: yellow), permanently satisfied (blue), permanently violated (red). Additionally, orange indicates that some constraints are conflicting, i.e., there is no possible sequence of future events that could satisfy all the conflicting constraints. The user can replay the whole trace automatically, manually process the events in the trace one by one, or jump to a specific event in the trace. This functionality is implemented using MP-Declare with Alloy[6] and MobuconLTL [16], the former of which accounts for the definition of data conditions in the input process model.

Inventory. The inventory is a novel artifact aimed at easing the management of the event logs and process models in use during the process analysis with RuM. As depicted in Fig. 2(f), all the files imported from the file system are retrievable from the inventory and intermediate results can be stored directly in the inventory as *snapshots*. Next to each snapshot, the inventory offers action buttons that can activate a related functionality. For example, event logs can be directly re-routed as inputs for discovery, process models can be sent to log generation, and so on.

3 User Evaluation

To assess the feasibility of RuM, we conducted a qualitative user evaluation. Table 1 shows the results of the post-study survey. The post-survey included the System Usability Scale (SUS) [5] and scales covering satisfaction, expectation confirmation, future use intentions, and usefulness [4]. The study involved eight participants: four BPM experts with little to no knowledge of DECLARE and four DECLARE experts. Overall, RuM was rated high on all scales. For example, RuM scored 81.875 on the

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

Table 1. Survey results represented as averages for both groups and overall. The SUS score ranges between 0 and 100, while the other scales range between 1 and 5.

	Overall	Declare experts	BPM experts
SUS	81.875	78.75	85
Satisfaction	4.5	4.5	4.5
Expectation	4.56	4.33	4.78
Future intentions	4.167	3.833	4.5
Usefulness	4.3125	4.25	4.375

SUS scale (69.69 is considered the average, while a score above 80 is considered to be good or excellent [13]). However, there was a significant difference between BPM and DECLARE experts. RuM was rated higher by BPM experts on all scales except satisfaction, which was rated as 4.5 by both groups. The largest differences between the two groups are SUS score (85 to 78.75) and future use intentions (4.5 to 3.833). This discrepancy can point towards Declare experts potentially being less sensitive to the improvements in the ease of use of DECLARE constraints. For a more detailed discussion on the study, we refer the interested reader to [3].

4 Conclusion

In this paper, we presented an overview of the functionalities currently available in RuM alongside the results of the post-study survey from a qualitative user evaluation. The results of the post-study survey show that RuM is well-usable for both novice and expert users. Furthermore, we believe that RuM provides all the essential and many additional functionalities necessary for process mining when using the DECLARE language and its extension MP-DECLARE.

For future work, we plan to explore the feasibility of developing a visual editor for MP-DECLARE models, so that elements can be added, deleted and modified in the graphical view directly. Another avenue for future work is to extend the current capabilities to support discovery of branched-DECLARE constraints [9]. Furthermore, we will investigate the integration of functionalities provided by other declarative frameworks for process analysis such as DCR Graphs [19], DPIL [20] and WoMan [11]. We will also continue working based on the feedback received during the user evaluation. While many improvements have already been made, we have not yet exhausted all of the ideas provided by the participants of the evaluation.

Acknowledgements. The work of A. Alman was supported by the Estonian Research Council (project PRG1226) and ERDF via the IT Academy Program. The work of C. Di Ciccio was partly supported by the Italian MUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza, and by the "SPECTRA" Sapienza research project.

References

1. 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 (PhD/Demos)*, pages 122–126, 2020.
2. A. Alman, C. Di Ciccio, D. Haas, F. M. Maggi, and J. Mendling. Rule mining in action: The RuM toolkit. In *ICPM DC/Tools*, pages 51–54, 2020.
3. A. Alman, C. Di Ciccio, D. Haas, F. M. Maggi, and A. Nolte. Rule mining with RuM. In *ICPM*, pages 121–128, 2020.
4. A. Bhattacherjee. Understanding information systems continuance: An expectation-confirmation model. *MIS Q.*, 25(3):351–370, 2001.
5. J. Brooke. SUS: a 'quick and dirty' usability scale. *Usability evaluation in industry*, page 189, 1996.
6. A. Burattin, F. M. Maggi, and A. Sperduti. Conformance checking based on multi-perspective declarative process models. *Expert Syst. Appl.*, 65:194–211, 2016.
7. C. D. Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi. Generating event logs through the simulation of Declare models. In *EOMAS@CAiSE*, pages 20–36, 2015.
8. M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst. Aligning event logs and declarative process models for conformance checking. In *BPM*, pages 82–97, 2012.
9. C. Di Ciccio, F. M. Maggi, and J. Mendling. Efficient discovery of Target-Branched Declare constraints. *Inf. Syst.*, 56:258–283, 2016.
10. C. Di Ciccio and M. Mecella. On the discovery of declarative control flows for artful processes. *ACM Trans. Manag. Inf. Syst.*, 5(4):24:1–24:37, 2015.
11. S. Ferilli. Woman: Logic-based workflow learning and management. *IEEE Trans. Syst. Man Cybern. Syst.*, 44(6):744–756, 2014.
12. C. W. Gunther and H. M. W. Verbeek. *XES - standard definition*, volume 1409 of *BPM reports*. BPMcenter.org, 2014.
13. P. T. Kortum and A. Bangor. Usability ratings for everyday products measured with the system usability scale. *Int. J. Hum. Comput. Interact.*, 29(2):67–76, 2013.
14. 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.*, 89:101482, 2020.
15. F. M. Maggi, C. Di Ciccio, C. Di Francescomarino, and T. Kala. Parallel algorithms for the automated discovery of declarative process models. *Inf. Syst.*, 74(Part):136–152, 2018.
16. 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*, pages 132–147, 2011.
17. F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst. Runtime verification of LTL-based declarative process models. In *RV*, pages 131–146, 2011.
18. F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, and P. J. Toussaint. Guided process discovery - A pattern-based approach. *Inf. Syst.*, 76:1–18, 2018.
19. M. Marquard, M. Shahzad, and T. Slaats. Web-based modelling and collaborative simulation of declarative processes. In *BPM*, pages 209–225, 2015.
20. S. Schönig and M. Zeising. The DPIL framework: Tool support for agile and resource-aware business processes. In *BPM (Demos)*, pages 125–129, 2015.
21. 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)*, pages 111–115, 2018.
22. T. Slaats. Declarative and hybrid process discovery: Recent advances and open challenges. *J. Data Semant.*, 9(1):3–20, 2020.
23. W. M. P. van der Aalst and M. Pesic. DecSerFlow: Towards a truly declarative service flow language. In *WS-FM*, pages 1–23, 2006.