# Rule Mining with RuM

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

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

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

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

Alexander Nolte
University of Tartu
alexander.nolte@ut.ee

***Abstract*—Declarative process modeling languages are especially suitable to model loosely-structured, unpredictable 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. However, using these techniques can become cumbersome in practical situations where different techniques need to be combined for analysis. In addition, the use of Declare constraints in practice is often hampered by the difficulty of modeling them: the formal expression of Declare is difficult to understand for users without a background in temporal logics, whereas its graphical notation has been shown to be unintuitive. In this paper, we present RuM, a novel application for rule mining that addresses the above-mentioned issues by integrating multiple Declare-based process mining methods into a single unified application. The process mining techniques provided in RuM strongly rely on the use of Declare models expressed in natural language, which has the potential of mitigating the barriers of the language bias. 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, Natural Language Processing**

## I. Introduction

Business Process Management (BPM) has become an integral part of how companies organize their workflows starting from the higher levels of management as recommended by ISO 9001 Quality Management Principles (especially principles 4 and 6) [1] to modeling and optimizing lower level processes through the use of various process mining techniques [2]. Process mining is the part of BPM that is focused on the analysis of business processes based on process execution logs (event logs). Process mining techniques can be based on two different approaches for representing process models that are used as their input and/or output: procedural process models or declarative process models.

Procedural process models aim at describing end-to-end processes and allow only for the process behavior that is explicitly specified in the model [3]. However, modeling step by step the entire control-flow of a business process can be cumbersome in some cases. For example, if the process is loosely-structured and has a high number of different paths and exceptions the model could become quickly unreadable. In these cases, it may be a better choice to use declarative process models that model the process as a set of rules that the process should follow. In this way, everything that is not constrained is allowed and several execution paths can be represented in a compact model.

In contrast with the multiple process mining applications available for working with procedural models, there are currently no similar applications for working with declarative models [4]. This lack of a comprehensive toolset can be considered as one of the main contributing factors of the relatively low adoption rate of Declare in the industry and has been named as one of the open research challenges in declarative process mining [5, RC7]. In addition to this, dealing with declarative process models is known to be difficult, especially for domain experts that generally lack expertise in temporal logics and, in most of the cases, find the graphical notation of Declare constraints unintuitive [6].

In this paper, we present RuM,[1] a novel process mining application that addresses these research challenges by integrating multiple Declare-based process mining techniques into a single unified application and by largely making use of natural language to express Declare constraints. With this tool, the user is not required to have any experience in temporal logics nor to be familiar with the graphical notation of Declare constraints, but can handle temporal properties using natural language statements. RuM implements process mining techniques based on MP-Declare [7] 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. Our aim was to (1) gain insights into how users from different backgrounds perceived the application and (2) identify means for improving it. In general, the application was well received and it was recognized to be timely and highly needed by all participants.

The remainder of this paper is structured as follows. Section II gives a short overview of the Declare language. Section III discusses the main design goals of RuM. Section IV gives an overview of the functionalities of RuM and lists the process mining techniques available in the application. Section V describes the user evaluation methodology and provides an overview of the evaluation results. Finally, Section VI concludes the paper and spells out directions for future work.

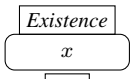
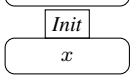
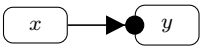
## II. Basics of Declare

Declare is a modeling language that uses a constraint-based declarative approach to model loosely-structured processes

[1] https://rulemining.org

Pre-print copy of the manuscript published by IEEE
(available at ieeexplore.ieee.org)

TABLE I
SOME DECLARE TEMPLATES

| Template | Explanation | Notation |
|---|---|---|
| Unary constraints | | |
| EXISTENCE($x$) | Activity $x$ occurs at least once per trace |   |
| INIT($x$) | Activity $x$ occurs at the beginning of every trace |   |
| Binary constraints | | |
| RESPONSE($x$,$y$) | If $x$ occurs, then $y$ must occur eventually after $x$ |   |
| CHAINRESPONSE($x$,$y$) | If $x$ occurs, then $y$ must occur immediately after $x$ |   |
| PRECEDENCE($x$,$y$) | $y$ occurs only if preceded by $x$ |   |

through behavioral constraints [8]. The language is grounded in linear temporal logic over finite traces [8], [9].

Declare is based on templates, which are parameterized temporal logic patterns and come endowed with a graphical notation to ease the depiction of process maps. To define process models using Declare, no knowledge of the underlying formal logic is required since Declare templates can also be expressed as natural language sentences. Declare constraints are concrete instantiations of templates obtained by replacing template parameters with real activities.

Table I shows some templates that will be used throughout the paper. Unary templates exert conditions on the occurrence of single activities. For example, EXISTENCE($x$) requires activity $x$ to occur in a trace. Binary templates predicate over pairs of activities. For instance, RESPONSE($x$,$y$) imposes that if $x$ occurs, then $y$ must eventually occur afterward. Binary templates partition their parameters into an activation and a target. The activation triggers the constraint, like the antecedent of a logical implication (e.g., $x$ for RESPONSE($x$,$y$)). The target is subject to the restriction exerted by the occurrence of the activation, like the consequent of a logical implication (e.g., $y$ for RESPONSE($x$,$y$)). Declare also includes *negative* versions of the binary templates: e.g., NOTRESPONSE($x$,$y$) states that if $x$ occurs, no $y$ can occur afterward.

A Declare model consists of a set of constraints that hold under logical conjunction. The behavioral semantics of every constraint can be represented by means of a finite-state automaton. A Declare model can thus be represented as a finite-state automaton stemming from the synchronous product of the automata of the single constraints [10].

More recently, the addition of conditions that predicate not only on activities and their control-flow, but also on data attributes and timestamps led to the definition of an extension

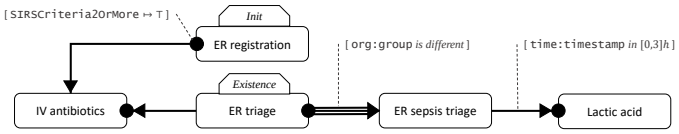



Fig. 1. An MP-Declare map

of standard Declare, namely MP-Declare [7]. Depending on the parameters on which data conditions insist, they are classified as *activation* conditions, *target* conditions or *correlation* conditions (the latter being exerted on both parameters). On the other hand, *time* conditions express constraints over the time distance between the activation and the target of a constraint.

Figure 1 depicts the graphical representation of an MP-Declare model (a map). The model is inspired by the analysis of a real-world event log[2] by Mannhardt et al. [11], [12] and refers to the healthcare process of handling patients affected by sepsis. The process begins with the emergency room (ER) registration. After that, if the activation condition on the Systemic Inflammatory Response Syndrome (SIRS) criteria attribute holds true, then intravenous (IV) antibiotics have to be administered. The IV antibiotics activity can only occur if ER triage was executed beforehand. Immediately after ER triage, the ER sepsis triage follows, but with the correlation condition that the actors carrying out the two activities differ (correlation condition on the `org:group` attribute). The analysis of lactic acid presence requires the ER sepsis triage to be run beforehand within 0 and 3 hours (time condition).

## III. DESIGN GOALS

In this section, we give a short overview of the main design goals of RuM. All the design goals are based on the combination of (1) the principle of "know your users" [13] and (2) the intended purpose of RuM [5, RC7]. We identified the target audience to be researchers and industry experts who may have varying levels of experience with Declare or declarative models in general. The main objective of RuM is to provide a comprehensive toolset for working with process mining techniques based on declarative models. Based on this, we identified the following 4 design goals.

First, the UI must have a low threshold for use and avoid an initial steep learning curve [14] especially since the target audience includes users that are not necessarily familiar with Declare. To achieve this, we decided to follow a minimalist visual design [15] and to divide the UI into different higher-level views based on the different tasks a user might want to carry on such as discovering a process model or checking the conformance of an event log.

Second, multiple different methods must be supported for each functionality (based on [5, RC7]). To achieve this, we selected and integrated multiple well-known Declare-based process mining techniques into RuM (Section IV). Additionally, we designed RuM in such a way that multiple process

[2] http://dx.doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

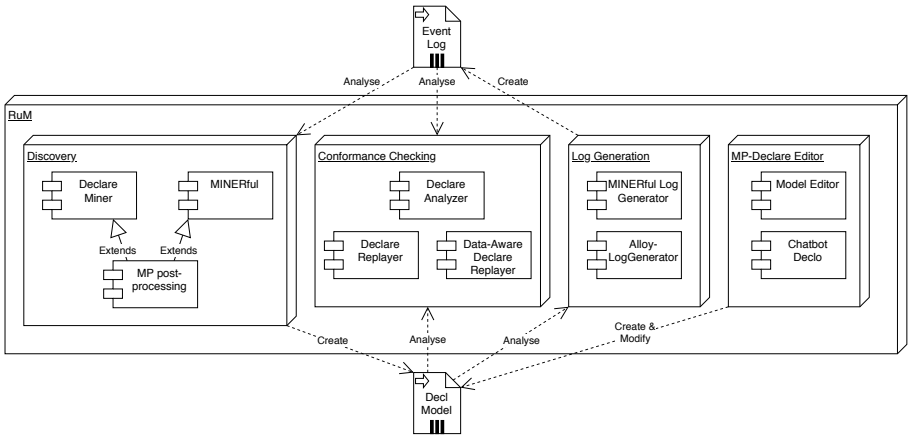



Fig. 2. Main functionalities of RuM

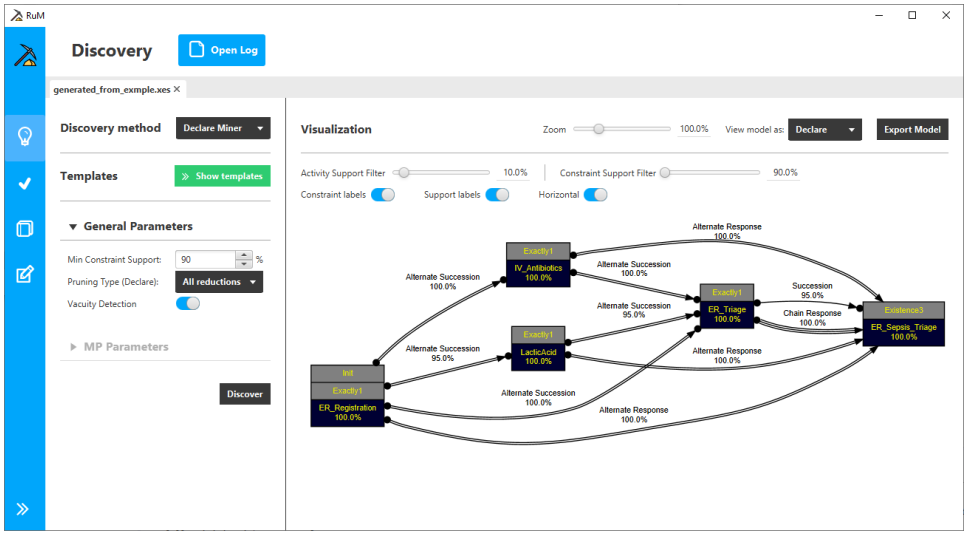



Fig. 3. Discovery UI

mining tasks can be started simultaneously. The user is thus free to navigate the rest of the application while a process mining task is ongoing.

Third, the functionalities in RuM should be easily reachable (principle of reachability [13]). To achieve this, we placed the input parameters of process mining methods into the same views where results are displayed or at most one click away via the use of slide-in panels. Additionally, we designed the UI of RuM in such a way that the results of two different process mining tasks are always at most two clicks away from each other (via the use of a side menu for navigating the main functionalities and a row of tabs for navigating the results).

Fourth, the parameters of different methods in the same functionality must be similar where possible (principle of consistency [13]). This is achieved by using the same input fields for all parameters that are common for different methods. If a parameter is specific to a selected method (for example process discovery techniques use different pruning approaches) then this parameter is explicitly labeled as method specific. The same counts for the results of different methods that must also be comparable. This was achieved by creating result views that have identical structure regardless of which method is used.

## IV. Functional Overview

RuM is the first software platform natively designed to analyze processes using a rule-based approach. To this 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 [16] for the event logs and *decl* [17] for the models. The diagram in Fig. 2 illustrates the software architecture of RuM with the components implementing its main functionalities: process discovery, conformance checking, log generation, and model editor. A demonstration video of these functionalities is available at: https://youtu.be/_6IMwR_SwaQ. In the following, we describe them in detail.

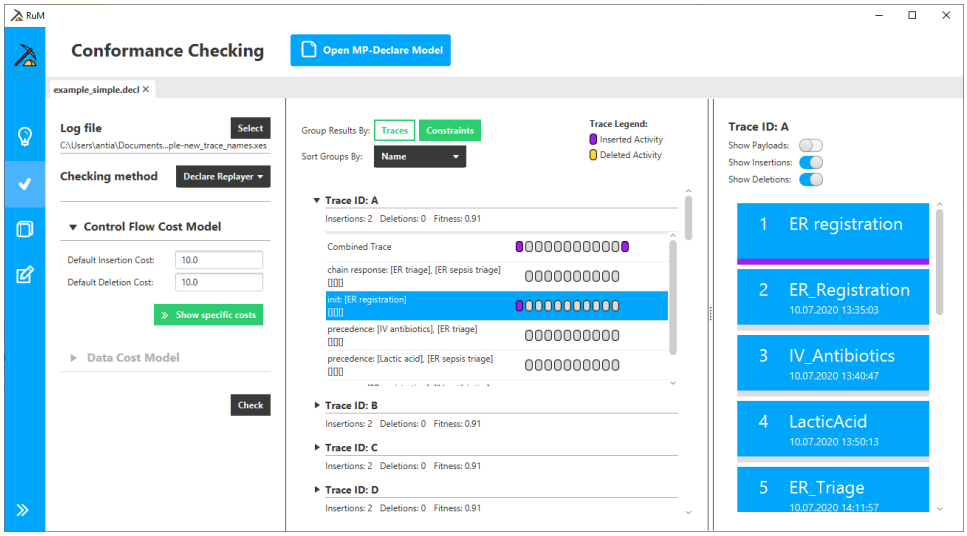



Fig. 4. Conformance Checking UI

### A. Discovery

Four methods are available for process discovery: Declare Miner [18], MINERful [19], MP-Declare Miner [20] and MP-MINERful. The MP variants add to the base mining algorithms a post processing step for discovering data conditions.[3]

The input parameters and the results are presented in the same view, as illustrated in Fig. 3. 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). In the remainder of this section, we describe them more in detail. We remark that those views support filtering based on activity support and constraint support thus providing the possibility for users to show/hide outlier behaviors.

*1) Declare View:* The Declare view represents the discovered model using the standard graphical notation for Declare constraints. Each activity in the model is represented as a single rectangle containing the activity name and the activity support (i.e., the percentage of log traces in which the activity occurs). Notice that the background of the activity rectangle is colored based on its support for immediacy of information conveyance. For constraints, it is possible to show or hide both the constraint name and the constraint support.

*2) Textual View:* The textual view is a model representation meant for users who are less familiar with the graphical syntax of Declare. The aim of the textual view is to describe the model using natural language sentences that are easy to understand without any prior knowledge of Declare.

*3) Automaton View:* The automaton view displays the discovered process model as a finite-state machine. This view is meant for users who are familiar with the formal semantics of Declare.

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

### B. Conformance Checking

There are three methods available for conformance checking: Declare Analyzer [7], Declare Replayer [21] and Data-Aware Declare Replayer.[4] The Declare Analyzer takes as input a model and an event log, and returns activations, violations, and fulfillments in the log of each constraint in the model. The Declare Replayer and the Data-Aware Declare Replayer report trace alignments. The Data-Aware Declare Replayer can also account for the data perspective. Both input parameters and results are presented in the same view (Fig. 4).

The conformance checking results are presented in groups. Each group represents the results for a specific trace or a specific constraint. Each group can be expanded to see the result details of that group. This allows users to explore the results at a high level of detail while also being relatively compact in terms of user interface. Notice that, if the result is a trace alignment, then it is also possible to show or hide both the activities that are inserted into the trace or removed from the trace as a result of the alignment.

### C. MP-Declare Editor

In order to provide a comprehensive toolset to work with Declare process models, RuM contains a model editor that supports not only standard Declare but also MP-Declare. The MP-Declare editor uses the *decl* file format to import and export 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 MP-Declare editor is presented in a single view (Fig. 5). Two slide-in panels are used, the first one for editing the activities and the second one for editing the attributes. Editing the constraints is done in a single table where each row corresponds to a single constraint. The entire model is also

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

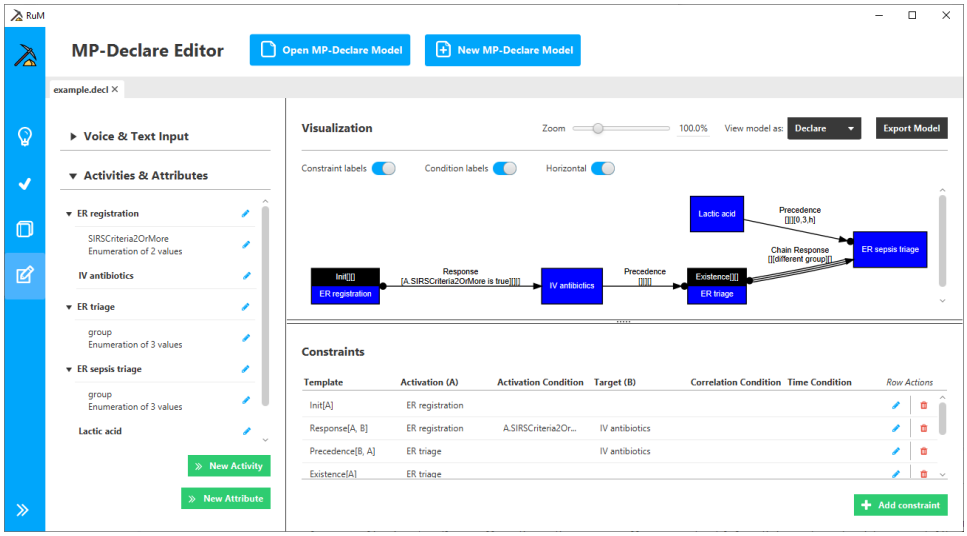



Fig. 5. MP-Declare Editor UI

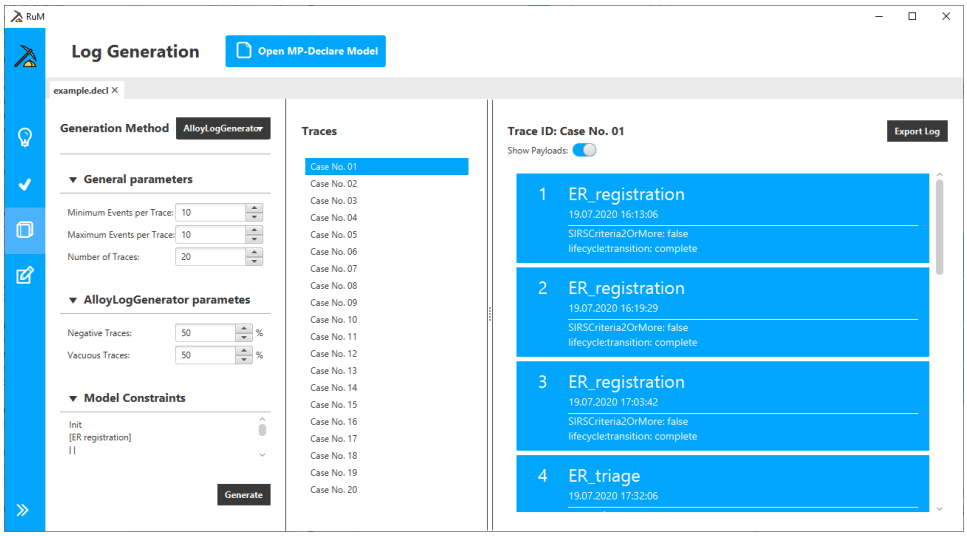



Fig. 6. Log Generation UI

represented visually in the same view and the visualization is updated on-the-fly as the user is editing the model. The used visualization devices are the same as those of the discovery panel, i.e., the model can be visualized using the standard Declare graphical notation, as text or in the form of an automaton.

Finally, in the editor, users can also add constraints and data conditions using natural language sentences. The sentences can be provided both by voice and text. This functionality is implemented as a simple chatbot named Declo [22].

### D. Log Generation

There are two log generation methods available in RuM: AlloyLogGenerator [17] and MINERful Log Generator [10]. The main difference between these methods from the user's standpoint is that the AlloyLogGenerator can also account for the definition of activation and correlation conditions in the input process model. Both the input parameters and the generated log are presented in a single view as illustrated in Fig. 6. Among other options, the user can specify the percentage of traces that trivially satisfy the constraints in the input model (i.e., traces that comply with the constraints because their activation never occurs) and the percentage of negative traces (i.e., traces that violate at least one of the constraints in the input model). The generated log can be exported in XES format.

## V. User Evaluation

To assess the feasibility of RuM, we conducted a qualitative user evaluation. Our aim was to (1) gain insights into how users from different backgrounds perceived the application and (2) identify means for improving it. In the following, we will write "(obs.)" to mark findings that are based on the observation of the participants while using the application.

### A. Study Setup

We selected eight process analysts as participants for our study. Four participants had little to no knowledge of Declare (B1 to B4), but worked in the BPM field (here called BPM experts), while the other four identified themselves as Declare experts (D1 to D4). We chose this differentiation to particularly study the potential differences related to their demands and perception about RuM. Prior to the study the participants were given access to a scenario,[5] some input files,[5] and the application itself. A common scenario was used for all participants in order to ensure the comparability of our findings.

The study was conducted via Skype by a team consisting of a *facilitator* and an *observer*, with the facilitator guiding the participant and the observer serving in a supporting role. It started with the facilitator introducing the study procedure and the application to the participants who were then asked to start RuM on their computer, share their screen and carry out the tasks based on the scenario provided earlier. Participants were encouraged to think aloud, to ask questions and to point out interesting aspects of the application during the test. Every test was video-recorded. After finishing the tasks, the facilitator conducted a short post-interview asking questions about the application in general and about the tasks where the participant appeared to have had difficulties.

Each study lasted between 45 and 60 minutes. After the end of the study, the participants received a link to a short post-survey[5] including the System Usability Scale (SUS) [23] and scales covering satisfaction, expectation confirmation, future use intentions, and usefulness [24]. To analyze the collected data, we focused on the video recordings, observations and follow-up interviews, using the post-surveys as an additional qualitative data point. We followed the affinity-diagramming method [25], which yielded 540 items that were divided into 10 main clusters and 111 sub-clusters in total.

### B. Study Scenario

For the study, we developed a scenario that involves common activities of a process analyst such as the discovery of a model from an existing event log and its validation and modification. We used the Sepsis Cases event log that is based on real-life treatment cases[2] as a basis and split it into a training and a test set. The participants were then first asked to use RuM to discover an initial model using the training set (Section V-D1). Afterward, they were instructed to check if the discovered model conforms to the test set using different conformance checking methods (Section V-D2). The participants were then asked to modify the discovered model based on the results of the conformance checking and some additional domain information (Section V-D3). Finally, the participants were asked to use the modified model to generate a new log (Section V-D4).

[5]The evaluation material is available at: https://git.io/JJIp4 (scenario); https://git.io/JJIpu (input files); https://git.io/JJLvB (post-survey).

TABLE II
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 |

### C. General Findings

Both groups of users found the UI of RuM to be usable as evident by statements such as *"nice interface"* (B1), *"I think it's a really nice tool, I really like it"* (D1), *"I think it's really cool"* (D4), *"I was impressed"* (D4). B3 also pointed out that everything in the user interface was understandable *"after clicking around for a few minutes"* (B3). These statements are underpinned by an average SUS score of 81.875 (a SUS score of 69.69 is considered average, while a score above 80 is considered to be good or excellent [26]). It was also pointed out that there is a need for an application like RuM *"I think it's very promising and also very much needed"* (B2), *"I think in the process mining community there was really need to freshen up Declare"* (D1).

However, there was a significant difference between BPM and Declare experts as evident by the post-survey results (Table II). 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 BPM and Declare experts are in the SUS score (85 to 78.75) and future use intentions (4.5 to 3.833). This discrepancy can point towards Declare experts being already used to existing tools for Declare-based process mining and potentially being less sensitive to the improvements in the ease of use of Declare constraints.

### D. Task-Specific Findings

A finding that was orthogonal to all tasks was the fact that, when it was needed to use the result of a section as input in another section, we deliberately did not mention exporting the results. While the process of exporting itself was not an issue, some participants (B2, D4) also expected the application to have quick ways to move files from one section to another with D4 stating *"its a bit strange that you have to export the model and then reopen it, the same one, in another tab"* (D4). Other findings (described in the following) were task specific.

*1) Discovery:* The ordering of templates in the template selection panel of the discovery section is based on template categories: unary templates, positive binary templates and negative binary templates. All BPM experts had difficulties in finding the correct templates from this panel, while the same was observed with only one Declare expert (*"ok, so they are not alphabetically sorted"*, D1). For example, B1 scrolled the template list from end to end multiple times before finding all the templates listed in the scenario (obs.). During the post-interview B4 suggested to add *"brief headings"* (B4) because

that would make it *"easier to quickly categorize it"* (B4). It was also mentioned that the large number of templates might be difficult to use (*"we have a lot of templates available which is maybe not straightforward"*, B3).

The second noticeable difference between the two groups was that three out of four BPM experts started working with the initial model that is discovered automatically by the tool with the default parameters (obs.), while our scenario explicitly asked them to use a different set of parameters. None of the Declare experts had the same issue (obs.) with only one mentioning the automatic discovery *"its interesting it immediately starts to discover something before I could set the parameters"* (D3). The problem was that, even if they changed the parameters, most of the BPM experts did not restart the discovery with the new parameters. This confusion might be related to filters that (differently from parameters) are applied on-the-fly to the discovered model (*"I did not do that because the map responded to some of the things I changed, for instance when I adapted the sliders"*, B2).

The model visualizations were considered good in general. For the Declare view, it was pointed out that the way unary constraints are displayed is *"a bit more intuitive than in all the papers"* (D2) and that the constraint template labels are useful to *"help explain the notation"* (D3). The textual view was considered a useful addition as evident by statements such as *"finally a textual version, I like it"* (D4) and *"it's quite good to have this written in text"* (B3). The automaton view, however, was generally considered to be more *"for theoretical people"* (D4).

Exporting the discovered model takes into account the currently opened model visualization and also the support filters. This means that if the user wants to export a model in *decl* format then the Declare view must be selected and only the parts of the model that are not filtered out with the support sliders are exported. However, not all participants assumed correctly that filters affect the exporting and some of them needed help to export the correct model (obs.).

*2) Conformance checking:* The implementation of the conformance checking feature in RuM was generally perceived well (*"it is presented in a pleasing way"*, D4, *"I like this conformance checking, well done!"*, D3). Both Declare and BPM experts had similar suggestions, comments and reported on similar issues related to the conformance checking task of our scenario.

One of the main differences of the conformance checking section with respect to the other sections is that it uses two input files (the model and the event log). This was solved in the user interface by treating the model as the main input of the section and the event log file as a parameter. During the evaluation this turned out not to be a problem. Only two participants (B2 and D2) paused for a moment before finding how to select the event log (obs.).

Switching between the original and aligned trace in the Declare Replayer appeared to be difficult for both groups with only B1 not needing any help (obs.). It was not obvious for most participants that "Show Insertions" and "Show Deletions" (Fig. 4) can be used for this purpose (*"I would not have guessed the meaning of that button"*, B2). Most participants needed some time or instructions from the facilitator to make the connection and to set the toggle buttons correctly (obs.). It was also noted that *"showing the deletion is like a negation of a negation"* (D3), which could also have been a source of confusion.

*3) Model editor:* The main difference between the two studied groups during model editing was that most of the Declare experts (D2, D3, D4) tried at first to edit the model by clicking on the visualization (obs.), which is not possible in RuM. Meanwhile, all the BPM experts used the visualization only to get an overview of the model and did not attempt to modify it (obs.). This could appear counter-intuitive since BPM experts are expected to be used to modifying graphical process models. However, this phenomenon could be related to the fact that the only existing Declare editor is a visual editor [27].

When editing the model, we asked the participants to remove a PRECEDENCE constraint. This turned out to be difficult for some of them (B4, D2, D3), because in the case of PRECEDENCE constraints the activities are presented in reverse order (target activity before the activation activity), when compared to other templates. This issue was specifically pointed out by statements such as *"Is this saying the same thing or is this saying the opposite thing?"* (B4), *"so this is confusing because it's the other way around"* (D2), *"ah right, the activation and target here are swapped"* (D3).

There were also some issues related to the syntax of data conditions. D1 and D2 attempted to use an equal sign instead of the word "is" for equalities (obs.), while B3 commented while entering the condition *"I hope this is the way to write it... is it so English like?"* (B3). D2 suggested adding a help button describing the syntax and D3 suggested that attributes in the model should be recognized by the editor while typing the data conditions.

*4) Log generation:* For the log generation, there were no noticeable differences between the two groups. Multiple participants (B1, B3, D2, D4) expected "Model Constraints" (Fig. 6) to be somehow functional (since it appears in the parameter panel). For example, B1 explored the constraint list for a bit before asking *"I don't have to put anything here?"* (B1) and B3 clicked multiple times on the constraints while exploring the generated log (obs.). In addition, multiple participants (B2, B3, B4, D4) attempted to click on the generated events when checking if an event was generated with the attributes required by the model (obs.). Instead, in the application, the attributes of the events in a trace can be shown all together using a toggle button (Fig. 6).

## VI. THREATS TO VALIDITY AND CONCLUSION

In this paper, we presented RuM, a comprehensive toolset for rule mining. Our evaluation provides indication that the tool is usable for novice and expert users. Novice users were particularly satisfied about the new look of Declare and about the effort made to improve the understandability of Declare

models, while Declare experts were more appreciating the fact that RuM collects the most widespread Declare-based process mining techniques in a single tool.

The goal of the study was to collect feedback and to evaluate the usability of RuM for individuals with different backgrounds. It was thus reasonable to conduct an in-depth qualitative study with selected participants from diverse backgrounds related to their knowledge and experience with Declare. Conducting a study with a small sample of participants is common because research has shown that the number of additional insights gained deteriorates drastically per participant [28].

There are, however, some threats to validity associated with this study design. First, we evaluate the use of a specific tool by specific individuals in a specific context over a limited period of time. Despite carefully selecting the participants and creating a setting that is close to how the tool would be commonly used, it is not possible to generalize our findings beyond our study context. In addition, the study was conducted by a team of researchers that might potentially interpret findings differently. We attempted to mitigate this threat by ensuring that observations, interviews and the analysis of the obtained data was collaboratively conducted. We also abstained from making causal claims, instead providing a rich description of the behavior and reported perceptions of participants.

For future work, we plan to continue developing RuM based on the feedback received during the user evaluation, focusing in particular on aspects that proved to be the most critical ones. Additionally, we plan to explore the feasibility of developing a visual editor for MP-Declare models, so that elements can be added/deleted/modified in the graphical view directly. We also plan to add an inventory of files that can be easily retrieved and used throughout all the sections of the tool. This will facilitate the construction of pipelines based on the analysis instruments provided by the application. Another avenue for future work is adding support for online analysis of event logs based on Declare taking inspiration from the works presented in [29] concerning online discovery and in [30] for process monitoring.

**Acknowledgements.** 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] L. Fonseca and J. P. Domingues, "ISO 9001:2015 edition- management, quality and value," *International Journal for Quality Research*, vol. 1, no. 11, pp. 149–158, 2017.

[2] W. M. P. van der Aalst, "Process mining: Overview and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 2, Jul. 2012.

[3] ——, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[4] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 686–705, 2019.

[5] T. Slaats, "Declarative and hybrid process discovery: Recent advances and open challenges," *Journal on Data Semantics*, vol. 9, no. 1, pp. 3–20, Mar 2020.

[6] C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, and B. Weber, "Understanding declare models: strategies, pitfalls, empirical results," *Software and Systems Modeling*, vol. 15, no. 2, pp. 325–352, 2016.

[7] 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.

[8] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*, 2006, pp. 1–23.

[9] G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on LTL on finite traces: Insensitivity to infiniteness," in *AAAI*, 2014, pp. 1027–1033.

[10] 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.

[11] F. Mannhardt and D. Blinde, "Analyzing the trajectories of patients with sepsis using process mining," in *RADAR+ EMISA@ CAiSE*, 2017, pp. 72–80.

[12] 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.*, vol. 76, pp. 1–18, 2018.

[13] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human-computer interaction*. Pearson Education, 2003.

[14] B. Myers, S. E. Hudson, and R. Pausch, "Past, present, and future of user interface software tools," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 1, pp. 3–28, Mar. 2000.

[15] J. Nielsen, "Ten usability heuristics," 1994, https://www.nngroup.com/.

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

[17] 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.

[18] 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.

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

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

[21] 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*, 2012, pp. 82–97.

[22] 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.

[23] J. Brooke, "SUS: a 'quick and dirty' usability scale," *Usability evaluation in industry*, p. 189, 1996.

[24] A. Bhattacherjee, "Understanding information systems continuance: an expectation-confirmation model," *MIS quarterly*, pp. 351–370, 2001.

[25] K. Holtzblatt, J. B. Wendell, and S. Wood, *Rapid contextual design: a how-to guide to key techniques for user-centered design*. Elsevier, 2004.

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

[27] M. Westergaard and F. M. Maggi, "Declare: A tool suite for declarative workflow modeling and enactment," in *BPM Demos*, 2011.

[28] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *INTERCHI*, 1993, pp. 206–213.

[29] A. Burattin, M. Cimitile, F. M. Maggi, and A. Sperduti, "Online discovery of declarative process models from event streams," *IEEE Trans. Serv. Comput.*, vol. 8, no. 6, pp. 833–846, 2015.

[30] 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, pp. 132–147.

This document is a pre-print copy of the manuscript
(Alman et al. 2020)
published by IEEE
(available at ieeexplore.ieee.org).

## References

Alman, Anti, Claudio Di Ciccio, Dominik Haas, Fabrizio Maria Maggi, and Alexander Nolte (2020). "Rule Mining with RuM". In: *ICPM*. Ed. by Boudewijn F. van Dongen, Marco Montali, and Moe Thandar Wynn. IEEE.

## BibTeX

```
@InProceedings{   Alman.etal/ICPM2020:RuM,
  author        = {Alman, Anti and Di Ciccio, Claudio and Haas, Dominik and
                   Maggi, Fabrizio Maria and Nolte, Alexander},
  title         = {Rule Mining with {RuM}},
  booktitle     = {ICPM},
  year          = {2020},
  crossref      = {ICPM2020},
  keywords      = {Rule Mining; Process Analytics Tool; Declarative Process
                   Models; Natural Language Processing}
}
@Proceedings{     ICPM2020,
  title         = {2nd International Conference on Process Mining, ICPM 2020,
                   Padua, Italy, October 5-8, 2020},
  year          = {2020},
  editor        = {Boudewijn F. van Dongen and Marco Montali and Moe Thandar
                   Wynn},
  publisher     = {IEEE}
}
```