DCR-JS: An Online Environment for Declarative Process Mining

Axel K.F. Christfort[1], Hugo A. López[2]

[1]Department of Computer Science, University of Copenhagen. Copenhagen, Denmark
[2]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark

Abstract

This paper presents an enhanced version of DCR-js, a modeling tool for DCR graphs, aimed at supporting novice users in the most common process mining tasks with declarative process models. Declarative models better capture business rules and knowledge-intensive processes but lack accessible tooling, limiting their industrial adoption. Our enriched DCR-JS framework supports modeling, simulation, discovery, conformance checking, and log generation with DCR graphs. The tool is actively used in academic settings, including BPM courses at the Technical University of Denmark, promoting an active learning environment for declarative process mining.

Keywords

Declarative Process Mining, DCR Graphs, Process Discovery, Conformance Checking, Event log Generation

1. Introduction

Process Mining is an established discipline with a market share of 871.6 million in 2023[1]. While most industrial players provide access to a range of different techniques for discovery and conformance, they do so based on *imperative process models*, such as Directly-Follows Graphs, Petri Nets, and BPMN variants. While valuable, imperative process models excel at describing process flows, processes where decisions are based on circumstantial information (for instance, compliance rules) are harder to represent with imperative models. Declarative models have been studied in the BPM literature with multiple notations, including CMMN, Declare, and DCR graphs [1]. Declarative process notations can discover the *business rules* affecting an organization [2]. Moreover, conformance checking techniques based on declarative models allow us to compare normative rules with logs [3] or other models [4].

Despite their benefits, the uptake of declarative process models is not comparable to their imperative counterparts, and to a large extent, declarative models have stayed within academic environments, with only a few examples of industrial adoption [5, 6]. We believe one of the reasons for this phenomenon is the lack of accessible tools to experiment with declarative process models. While some tools exist [7, 8], they either require academic licenses, are closed-source, or necessitate complex installation processes.

This paper departs from our DCR-JS [9] modelling tool for DCR graphs, enriching it to create a mature framework for modelling, simulation, discovery, conformance, and log generation based on DCR graphs. The framework supports the most important use cases in declarative process mining. It includes state-of-the-art implementations used as a benchmark in the Process Discovery Challenge[2]. DCR-JS is currently being used in the teaching of two BPM courses at the Technical University of Denmark, supports the discovery of a large set of benchmarking event logs, and includes bug fixes and features contributed by the students themselves.

Document Structure Section 2 introduces related work. Section 3 introduces DCR graphs via a modelling example. Section 4 introduces the innovations of the tool. Section 5 comments on the maturity

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 23rd International Conference on Business Process Management (BPM 2025), Seville, Spain, August 31st to September 5th, 2025.
axel@di.ku.dk (A. K.F. Christfort); hulo@dtu.dk (H. A. López)
http://lopezacosta.net/ (H. A. López)
0000-0001-5681-5936 (A. K.F. Christfort); 0000-0001-5162-7936 (H. A. López)

[1]https://www.gartner.com/en/documents/5633491
[2]https://icpmconference.org/2024/process-discovery-contest

of DCR-JS. Section 6 provides links to source code and documentation, and Section 7 concludes.

2. Related Work

DCR-JS is part of the ecosystem of declarative process modelling tools, including RUM [8], Declare-js [10], and EasyDeclare [11]. While [10] and [11] focus on modelling, they do not consider process mining tasks. RUM [8] will be the closest tool from the Declare toolset, with an intended use as a standalone application. The set of DCR-based tools includes the DCR Portal [7], DCR4Py [12], and our previous version of DCR-JS [9]. These tools are complementary to us, being the DCR process portal an authoring tool for executable DCR models, and DCR4Py [12] is a tool for process mining of DCR graphs for process analysts with coding skills. DCR-JS focus is to render the typical declarative process mining tasks accessible to novice users, with no installation or coding skills required.

3. A Primer on DCR graphs

We use a simple DCR graph for a booking process in Figure 1 to illustrate the main modelling constructs. For the formal definition of the constructs used, please refer to [1], and for a full overview of the language set in DCR graphs, see [13]. A DCR can be seen as a collection of *events* linked by *relations*. Each event has a *marking*, denoting whether the event has been executed, is pending, or has been excluded. Relations impose constraints on accepting traces. The type of constraint determines the accepted behaviour. A condition relation () defines precedence, and a response () will determine obligatory consequences. An exclusion () will remove elements from the environment that can later be included via inclusion relations (). Milestone relations () block activities from being executed if their included dependencies are pending. Finally, relations can be composed via simple conjunction, and a nesting event will denote the application of the rules to the outermost event to its innermost constituents.

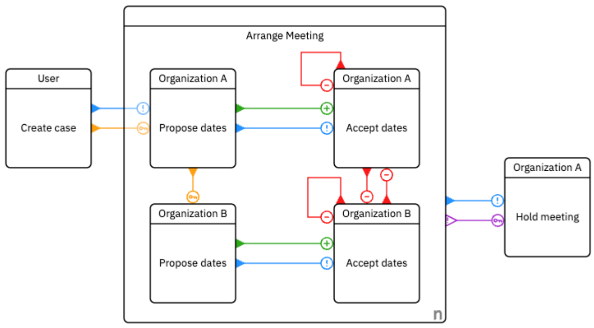

Figure 1: Booking Process in DCR Graphs

4. Innovations of the Tool

This version of DCR-JS includes major innovations in its five existing modules, illustrated in Figure 2:
Modelling: This is the only prior module in DCR-JS [9]. The current version includes support for three DCR graphs process notations, including the formal notations [1], as well as support for commercial[3] and notations optimized for semantic-transparency [14], and accessibility options for color-blind users. Moreover, we provide automatic layout capabilities, including alternatives to reduce the graphic complexity via automated event nestings [15], as well as the capability to do test-driven modeling with open test cases [16] to ensure compliance throughout modeling and extension.
Discovery: It generates DCR graphs from XES event logs. The module supports the introduction of multiple DCR discovery algorithms, including DisCoveR [17] and the rejection miner [2], as well as noise-filtering capabilities [18].
Simulation: It supports the step-based simulation of DCR graphs, providing user feedback on the execution of disabled events, and allowing the generation of non-conforming traces for testing. User-based simulations can be stored as XES logs.

[3] https://dcrsolutions.net/

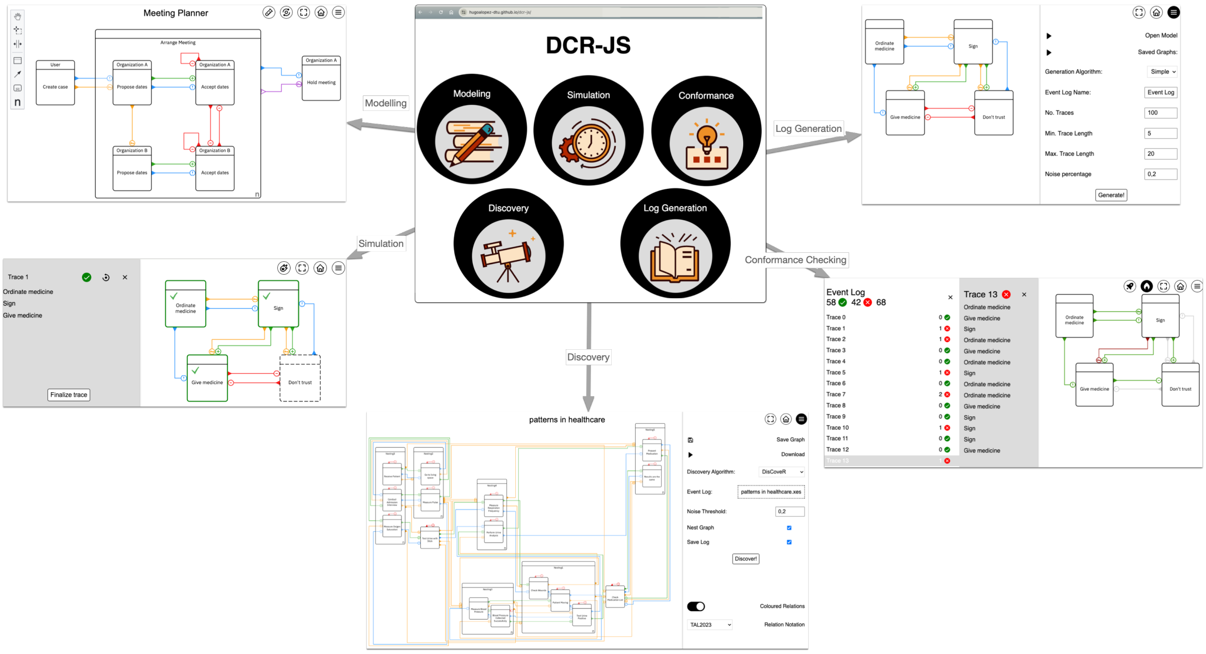

Figure 2: DCR-JS and its support for declarative process mining tasks

Conformance Checking: This new mode supports rule-based and alignment-based conformance checking for DCR graphs [12, 19]. Visualization capabilities are based on Gestalt principles, supporting the user in understanding the rules that have been triggered and fulfilled, violated, or not activated.
Log Generation It allows the generation of XES event logs from a DCR graph. To support the generation of non-conforming traces, the mode supports the automatic injection of noise levels for the traces generated.

4.1. DCR-JS Architecture

DCR-JS is composed of three main modules: First, a main interactive web-based tool supports modelling, simulation, and manipulation of DCR graphs. Second, the original DCR-JS [9] editor provides visualization and modelling capabilities. Finally, the DCR engine performs model updates and process mining tasks. Figure 3 shows its main modules.

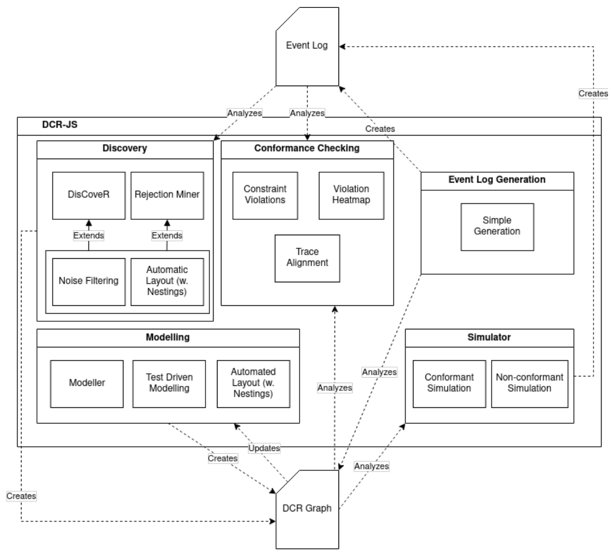

Figure 3: Component Diagram for DCR-JS

5. Maturity of the tool

DCR-JS has been developed since 2023 [9] and has grown with major re-engineering efforts and software engineering projects at the Technical University of Denmark and Copenhagen University. It is currently used in two MSc courses at DTU, with students contributing to the tool. DCR-JS supports the following use cases:
Process of Process Mining: Discover a DCR graph from an event log, then use the modeler to modify the graph (add/remove/change the type of constraints), and use conformance checking to identify the

Event Log	Size (MB)	Cases	Variants	Events	Time (ms)	Constraints	Graph
CAiSE'17	192,6	100.000	346	894.708	367	11	link
Sepsis log	5,4	1.050	846	15.214	419	91	link
BPIC'20: RFP	15,2	6.886	89	36.796	798	196	link
BPIC'13, incidents	39,6	7.554	2.278	65.533	1.394	7	link
Large reviewer log	69,3	10.000	4.118	154.240	2.561	84	link
BPIC'12	74,1	13.087	4.366	262.200	3.135	156	link
BPIC'17, Offer log	109,8	42.995	16	193.849	5.051	30	link
Hospital Billing Log	174,3	100.000	1.020	451.359	8.622	115	link
Bank Transaction Process	162,1	10.000	10.000	678.864	19.524	1375	link
BPIC'19	739,2	251.734	11.973	1.595.923	OoM	NA	NA

Table 1
Discovery Performance

impact of the changes. Final models can be used as part of a PAIS.
Iterative Modelling/Redesign: Here, the task is to build a process model that abides by a certain set of constraints. Here, starting in the modeling tool, the modeler adds constraints and verifies them in a step-wise simulator to generate characteristic logs with compliant/violated traces, and iteratively adding new constraints or events. Once the modeler is satisfied, they can use the event log generator to characterize scenarios used for testing in a later process evolution phase.
Test-Driven Modelling: Starting with either a set of events or an existing DCR graph, Test-Driven Modelling defines several open test cases to ensure compliance of the model before adding constraints or extending it. Each test case consists of a partial trace, a context, and a polarity. A positive test ensures that there exists a trace in the model that, when projected onto the context, is exactly the test trace. A negative test ensures that there is no such trace in the model. By having these partial traces and contexts, you ensure that old test cases still work, even if the model is extended.
Scalability: The tool has an academic scope, and the design decision of making it easily accessible has ruled over scalability. DCR-JS is constrained by the memory limitations of the browser being used. Table 1 reports process discovery times over a range of classical event logs[4], showing acceptable performance in most of them. The largest overhead comes from log parsing, followed by discovery and layout. Larger logs (e.g., Road Traffic Fines, BPIC'19) would represent a bottleneck, and standalone implementations [12] should be used instead. Future extensions of the tool will consider the inclusion of asynchronous threads and standalone versions to remove memory limitations.

6. Availability

DCR-JS can be accessed at https://hugoalopez-dtu.github.io/dcr-js/ and does not require installation. The code and the user manual are available at https://github.com/hugoalopez-dtu/dcr-js under the MIT license. A screencast documenting the most important use cases is available at http://tiny.cc/ya1o001.

7. Conclusion

DCR-JS is designed to facilitate the introduction of process modelling and process mining techniques for novice practitioners interested in learning about the benefits of declarative process modelling. In future work, we want to extend the support to models including time, data, and object-centric variants.

Acknowledgments

Thanks to A. Andaloussi, K. de Place, M. Al-Helo, J. Nørgaard, T. Slaats, and D. Trinh for their suggestions and/or contributions to the tool. This work is supported by the research grant "Center for Digital

[4]Tests executed on an Apple M1 Pro with 32 GB RAM, running Mac OS 15.5, and Google Chrome 138.0.7204.93

ComplianceE (DICE)" (VIL57420) from VILLUM FONDEN, and by the Innovation Foundation project "Explainable Hybrid-AI for Computational Law and Accurate Legal Chatbots" 4355-00018B XHAILe.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly as a grammar and spell-checker.

References

[1] T. T. Hildebrandt, R. R. Mukkamala, Declarative event-based workflow as distributed dynamic condition response graphs, arXiv preprint arXiv:1110.4161 (2011).
[2] T. Slaats, S. Debois, C. O. Back, A. K. F. Christfort, Foundations and practice of binary process discovery, Information Systems 121 (2024) 102339.
[3] A. Burattin, F. M. Maggi, A. Sperduti, Conformance checking based on multi-perspective declarative process models, Expert systems with applications 65 (2016) 194–211.
[4] H. A. López, S. Debois, T. Slaats, T. T. Hildebrandt, Business process compliance using reference models of law, in: FASE, 2020, pp. 378–399.
[5] E. Marengo, P. Dallasega, M. Montali, W. Nutt, M. Reifer, Process management in construction: Expansion of the bolzano hospital, in: BPM Cases, Springer, 2018, pp. 257–274.
[6] T. T. Hildebrandt, et al., Ecoknow: Engineering effective, co-created and compliant adaptive case management systems for knowledge workers, in: ICSSP, 2020, pp. 155–164.
[7] M. Marquard, M. Shahzad, T. Slaats, Web-based modelling and collaborative simulation of declarative processes, in: BPM, Springer, 2015, pp. 209–225.
[8] A. Alman, C. Di Ciccio, F. M. Maggi, M. Montali, H. van der Aa, Rum: declarative process mining, distilled, in: BPM, Springer, 2021, pp. 23–29.
[9] L. K. Tamo, A. Abbad-Andaloussi, D. M. T. Trinh, H. A. López, An open-source modeling editor for declarative process models, in: CoopIS 2023, CEUR-WS, 2023, pp. 1–5.
[10] S. Nagel, E. Amann, P. Delfmann, declare-js: A web-based viewer and editor for declarative process models., in: ICPM Doctoral Consortium/Demo, 2023.
[11] G. Blasilli, L. S. Ferro, S. Lenti, F. M. Maggi, A. Marrella, T. Catarci, Edd: A web-based editor for declarative process models using easydeclare (2024).
[12] S. V. Hermansen, R. Jónsson, J. L. Kjeldsen, T. Slaats, V. P. Cosma, H. A. López, DCR4Py: A PM4Py library extension for declarative process mining in python, in: ICPM, CEUR-WS, 2024.
[13] H. A. López, V. D. Simon, How to (re) design declarative process notations? a view from the lens of cognitive effectiveness frameworks, in: PoEM 2022, CEUR-WS, 2022.
[14] D. M. T. Trinh, A. Abbad-Andaloussi, H. A. López, On the semantic transparency of declarative process models: the case of constraints, in: CoopIS, Springer, 2023, pp. 217–236.
[15] V. P. Cosma, A. K. F. Christfort, T. T. Hildebrandt, X. Lu, H. A. Reijers, T. Slaats, Improving simplicity by discovering nested groups in declarative models, in: CAISE, Springer, 2024, pp. 440–455.
[16] A. K. Christfort, V. P. Cosma, S. Debois, T. T. Hildebrandt, T. Slaats, Static and dynamic techniques for iterative test-driven modelling of dynamic condition response graphs, Data Knowl Eng 157 (2025) 102413.
[17] C. O. Back, T. Slaats, T. T. Hildebrandt, M. Marquard, Discover: accurate and efficient discovery of declarative process models, International Journal on STTT 24 (2022) 563–587.
[18] A. K. F. Christfort, S. Debois, T. Slaats, Improving declarative process mining with a priori noise filtering, in: BPM, Springer, 2022, pp. 286–297.
[19] A. K. F. Christfort, T. Slaats, Efficient optimal alignment between dynamic condition response graphs and traces, in: BPM, Springer, 2023, pp. 3–19.