An Open-Source Modeling Editor for Declarative Process Models

Lucien Kiven Tamo[1,*], Amine Abbad-Andaloussi[2,*], Dung My Thi Trinh[1,*] and Hugo A. López[1,*]

[1]Technical University of Denmark, Richard Petersens Plads, 321, 2800 Kgs. Lyngby, Denmark
[2]University of St Gallen, St Gallen, Switzerland

Abstract

This paper presents an open-source modeling environment for Declarative Process Models. Traditionally, process models have described rigid structures, where re-work, process variants, and alternatives are difficult to represent. The Dynamic Condition Response (DCR) Graphs notation is a declarative process modeling notation that enables the description of processes with a high level of flexibility, using behavioral constraints to allow only compliant executions. The DCR-js editor is an open-source web-based editor for DCR graphs for academic use that enables the process management community to interact with declarative process models. As part of the innovations of the tool, DCR-js provides an alternative, semantic-transparent representation of declarative process models. Its web-based interface makes it ideal to be used as a component in tool-based experiments on cognitive aspects of modeling business processes. With DCR-js, we expect to render declarative process models more accessible to novice and academic users.

Keywords

DCR Graphs, Declarative Process Models, Modeling Editor

1. Introduction

In the pursuit of continuous improvement and innovation, organizations have historically turned to process modeling techniques as indispensable tools for optimizing efficiency and productivity while ensuring customer satisfaction. Traditionally, process modeling has served as a crucial method for capturing and depicting how multiple agents in an organization interact to achieve objectives by an ordered execution of tasks. These well-established techniques, commonly referred to as *imperative processes*, have proven invaluable in facilitating a structured approach to handling routine tasks within an organization [1]. However, as the business landscape evolves and complexity grows, there is a rising recognition that imperative process modeling may have limitations in addressing the dynamic and adaptive nature of modern operations. To bridge this gap, *declarative process modeling* was proposed. Unlike its predecessor, declarative process

Proceedings of the Demonstration Track at International Conference on Cooperative Information Systems 2023, CoopIS 2023, Groningen, The Netherlands, October 30 - November 3, 2023

*Corresponding authors.
hulo@dtu.dk (H. A. López)
http://lopezacosta.net (H. A. López)
0000-0001-5162-7936 (H. A. López)

modeling focuses on leaving an open execution canvas, limiting the possible executions via behavioral constraints between activities [1].

One valuable notation that falls under the declarative paradigm is the Dynamic Condition Response (DCR) [2]. In comparison to other declarative process modeling notations like DECLARE [3] and CMMN [4], DCR has gained popularity in the process modeling community with a limited but solid base of users and it is actively being taught in many Scandinavian universities. DCR provides a graphical formalism for modeling, analyzing, and optimizing dynamic processes, that has evolved over years to capture complex behavioral patterns including control flows [2], subprocesses [5], data [6], time [7], and message-passing constraints [8]. Over the past years, DCR has also benefited from a large array of empirical studies aiming at improving the modeling and comprehension of declarative process models [9, 10]. However, despite their flexibility, there is a considerable lack of adoption of declarative process models compared to their imperative counterparts. Several factors may contribute to the lack of adoption. One prominent obstacle is the absence of adequate open-source tools tailored to facilitate DCR graph modeling and rendering. Existing solutions are intended for expert users, focus on process execution use cases, and live under closed-use licenses, thus limiting the accessibility of DCR graphs for a larger audience of novices in declarative process languages. Moreover, without simpler tools intended for novice users, potential users may find the learning curve steep and, as a result, be discouraged from adopting novel declarative modeling techniques.

This paper introduces DCR-js, an open-source DCR graphs editor intended for novice users. Our primary objective is to render declarative process models easier to access for a larger population of users, mainly interested in different use cases than the ones covered by commercial distributions, namely teaching, researching and process mining. DCR-js draws inspiration from existing web-based process editors like bpmn-js[1]. We leverage the capabilities of the diagram-js[2] library (empoyering bpmn-js) to construct a process editor specifically designed for DCR graphs. Our focus remains on achieving comprehensive support for the standard DCR graphs modeling syntax but introduces novel notations intended to solve some of the shortcomings of the existing notation explored in the literature [11, 12]. The alternative representations aim at supporting novice users in the adoption of DCR graphs while still being compliant with the standard notation used by DCR experts. The remainder of this paper is structured as follows: Section 2 presents a brief background on DCR graphs. Then, Section 3 provides an overview of the DCR-js features. Afterward, Section 4 discusses the maturity of the DCR-js editor. Finally, Section 5 concludes the paper and provides an overview of the planned future work.

2. Background on DCR

Since its inception in 2011 [2], the DCR process modeling notation has known many changes and additions to the language [11]. The fragment of the DCR language implemented in the tool includes control flow operators and roles [2], nesting [5] and subprocesses [7], and constitute the most common features used for teaching DCR graphs. They are illustrated in Figure 1.

The DCR notation includes events, relations, and markings. An **Event** describes the possible

[1] https://github.com/bpmn-io/bpmn-js
[2] https://www.npmjs.com/package/diagram-js

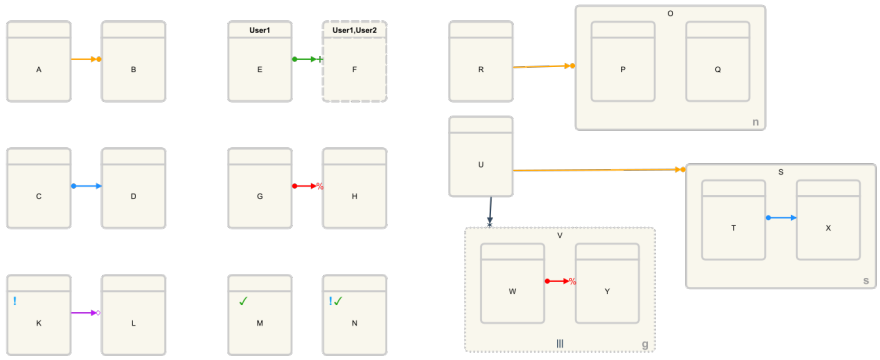

Figure 1: DCR Graph: notation example.

activities existing in a process. Events can take the form of atomic occurrences (e.g. *A* in Fig. 1) or be grouped in event collections (e.g. *O*). Additionally, events can be associated with zero or multiple roles (e.g. *E* and *F*). Directed **Relations** constrain the execution of events. The notation implemented corresponds to the operators in [7], and includes 6 types of relations: (*i*) conditions (→•), (*ii*) includes (→+), (*iii*) responses (•→), (*iv*) excludes (→%), (*v*) milestones (→), and (*vi*) spawns (→*). A *condition* from *A* to *B* limits the execution of *B* until *A* is executed or excluded. A *response* from *C* to *D* makes *D* pending if *C* is executed. An *include* from *E* to *F* makes *F* included if *E* is executed. The *exclude* does exactly the opposite of an include. A *milestone* from *K* to *L* blocks the execution of *L* if *K* is pending for execution.

Event states are represented using **markings** which determine if an event can be executed or not. The set of markings comprises included/excluded events (differentiated via solid/dashed borders, e.g., *E* and *F* respectively), executed/not executed events (e.g. those having the ✓ check mark e.g. *M*), and pending/non-pending events (e.g. those having the ! symbol e.g. *K*). Markings are compositional (e.g. *N*). **Event collections** consist of a stateless nesting structure (e.g. *O*) which is a graphic economy operator, reducing the complexity of the model by allowing modelers to apply a single constraint to multiple events [5]. Stateful single-instance and multi-instance subprocesses (e.g. *S*, *V* respectively) create copies of each of the enclosed events and relations at runtime [7]. A multi-instance sub-process is bound to normal events via the *spawn* relation.

3. Overview of Tool Features

The DCR-js editor is a web-based framework adapted to recent web browsers. Figure 2 shows the graphical user interface of our application while its principal components (i.e., Canvas, Palette Panel, Export/Import Panel, Context Pad and Popup Menu, Settings Panel) are discussed in the following paragraphs:

Canvas. When opening the DCR graph modeling tool, users are presented with the canvas as the central area of interaction. The canvas serves as the workspace where users can construct and modify DCR graphs. It is designed to support a range of functionalities, including drag and drop of modeling constructs, zooming in and out, repositioning, resizing and labeling.

Palette Panel. The palette panel provides users with a collection of tools to construct and modify DCR graphs. It consists of three tools for manipulation (hand tool, lasso tool, and space

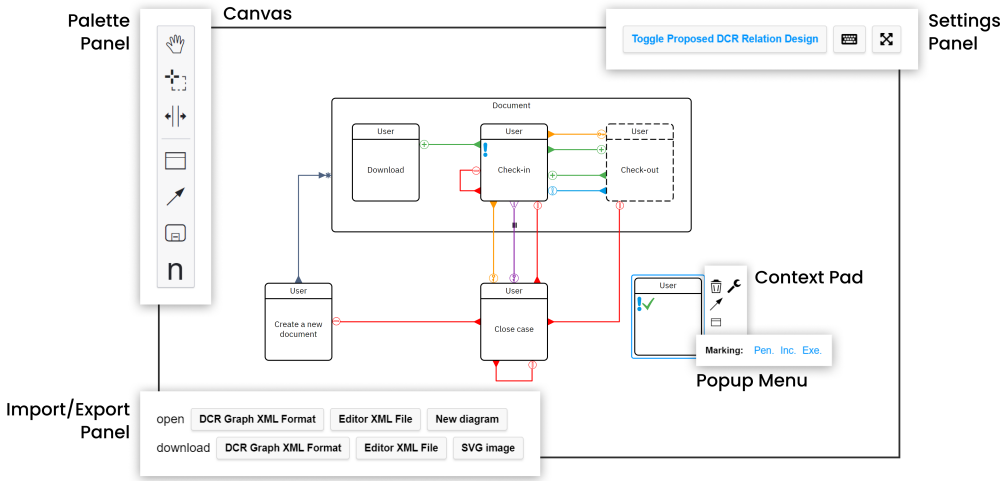

Figure 2: Overview of the DCR-js editor with the new semantic-transparent DCR notation activated [12].

tool) and four pre-defined elements (Events, relations, nesting and sub-processes, and labels).

Import/Export Panel. The import/export panel implements the persistency layer of the tool, adding cross-platform compatibility with the XML definition schema of DCR graphs [13].

Context Pad and Popup Menu. The context pad is an interactive menu that appears when the user selects an element on the canvas. It allows connecting, configuring, and deleting elements. Configuring an element can be done through the wrench button which opens a popup menu with element-specific functionalities. For instance, If the element is an event, the popup menu shows event modifiers, that encode the markings for pending, included, and executed events, whereas, if the element is a relation, the popup menu allows changing its type.

Settings Panel. The settings panel consists of three buttons: “Toggle Proposed DCR Relation Design”, “Keyboard”, and “Fullscreen”. The “Toggle Proposed DCR Relation Design” allows users to model DCR graphs using the alternative, semantic-transparent notation for DCR graphs derived following our recent empirical study investigating how to model DCR relations with higher semantic transparency [12]. The “Keyboard” and ”Fullscreen” buttons provide usability features to facilitate the interaction with the editor on larger screens and using the keyboard.

4. Maturity

The DCR-js editor is implemented in JavaScript and tested on Google Chrome and Microsoft Edge browsers for compatibility. The project is available at the GitHub repository https://github.com/hugoalopez-dtu/dcr-js. The editor is available online at https://hugoalopez-dtu.github.io/dcr-js/ . Additionally, the features of DCR-js are showcased in the following video https://youtu.be/1AQ6YdtgUUA. The editor supports a wide range of DCR graph elements, allowing us to model the control flow of business processes, much like the well-established commercial-grade but closed-sourced tool the DCR Graphs Portal[3]. However, unlike the DCR

[3]See https://www.dcrgraphs.net/

Graphs Portal, our editor provides users with an open-source framework that allows for the extension of the tool (e.g., the implementation of a new representation of DCR relations based on the proposal in [12]). Furthermore, the editor is conceived with interoperability in mind, and users can export DCR models in the XML format using the schema described in [13].

5. Conclusion and Future Work

The DCR-js editor aims to support teaching and research on declarative process models. In future work, we expect to extend the editor with further language capabilities such as data, time, and communication flows, as well as analysis, simulation, and verification capabilities.

References

[1] M. Reichert, B. Weber, Enabling flexibility in process-aware information systems: challenges, methods, technologies, volume 54, Springer, 2012.
[2] T. Hildebrandt, R. R. Mukkamala, Declarative event-based workflow as distributed dynamic condition response graphs, arXiv preprint arXiv:1110.4161 (2011).
[3] M. Pesic, H. Schonenberg, W. M. Van der Aalst, Declare: Full support for loosely-structured processes, in: EDOC, IEEE, 2007, pp. 287–287.
[4] OMG, Case Management Model and Notation, Version 1.1, 2016. URL: http://www.omg.org/spec/CMMN/1.1.
[5] T. Hildebrandt, R. R. Mukkamala, T. Slaats, Nested dynamic condition response graphs, in: FSEN, Springer, 2011, pp. 343–350.
[6] R. Strømsted, H. A. López, S. Debois, M. Marquard, Dynamic evaluation forms using declarative modeling, in: BPM (Dissertation/Demos/Industry), volume 2196 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 172–179.
[7] S. Debois, T. Hildebrandt, T. Slaats, Replication, refinement & reachability: complexity in dynamic condition-response graphs, Acta Informatica 55 (2018) 489–520.
[8] T. Hildebrandt, H. A. López, T. Slaats, Declarative choreographies with time and data, in: BPM (Forum), volume 490 of *LNBIP*, Springer, 2023, pp. 73–89.
[9] A. Abbad Andaloussi, C. J. Davis, A. Burattin, H. A. López, T. Slaats, B. Weber, Understanding quality in declarative process modeling through the mental models of experts, in: BPM, Springer, 2020, pp. 417–434.
[10] A. Abbad-Andaloussi, A. Burattin, T. Slaats, E. Kindler, B. Weber, Complexity in declarative process models: Metrics and multi-modal assessment of cognitive load, Expert Systems with Applications 233 (2023) 120924.
[11] H. A. López, V. D. Simon, How to (re) design declarative process notations? a view from the lens of cognitive effectiveness frameworks, in: PoEM, CEUR-WS, 2022.
[12] D. M. T. Trinh, A. Abbad-Andaloussi, H. A. López, On the semantic transparency of declarative process models: The case of constraints, in: COOPIS 2023, (Accepted for publication), Springer, 2023.
[13] T. Slaats, R. R. Mukkamala, T. Hildebrandt, M. Marquard, Exformatics declarative case management workflows as dcr graphs, in: BPM, Springer, 2013, pp. 339–354.