CEUR-WS.org/Vol-2469/ERDemo02.pdf

chor-js: A Modeling Framework for BPMN 2.0 Choreography Diagrams

Jan Ladleif, Anton von Weltzien, and Mathias Weske

Hasso Plattner Institute, University of Potsdam, Germany
{jan.ladleif,mathias.weske}@hpi.de, anton.weltzien@student.hpi.de

Abstract. Choreography diagrams were introduced with the release of Business Process Model and Notation (BPMN) 2.0 to model inter-organizational business processes, for example in the context of contractual agreements. They have, however, failed to attract a degree of tooling support comparable to their process and collaboration counterparts. With `chor-js`, we alleviate this deficit by providing a web-based, open-source choreography modeling framework based on `bpmn-js`. `chor-js` aims to be an extensible, intuitive and easy-to-integrate choreography diagram modeler faithful to the BPMN 2.0 standard and targeted at researchers and users alike.

Keywords: BPMN · Choreography Diagrams · Modeler.

1 Introduction

In its second version, the BPMN standard was extended by choreography diagrams [3]. Choreography diagrams abstract from collaboration diagrams in that they hide private orchestration details and focus on the participants' interactions only. Despite the general popularity of BPMN, choreography diagrams did not reach a similar degree of adoption in industry as their process counterparts [2].

In research, however, choreography diagrams emerge regularly, most recently due to their utility in model-driven smart contract development for blockchain applications [1, 4]. In fact, Mendling et al. foresee that "[the] whole area of choreographies may be re-vitalized by [blockchain] technology" [2]. A serious roadblock towards this goal is the relative lack of tools and frameworks targeting choreography diagrams, though: While a number of major BPMN modeling platforms do provide basic support, they suffer from various issues like non-extensibility, a lack of maintenance, or severe bugs (see Sect. 3.1). Thus, the conceptual potential of choreography diagrams remains largely untapped.

With `chor-js`, we want to contribute to the modeling community by enabling developers to properly implement choreography diagrams into their tooling and research in a modern, easy-to-use and highly extensible fashion. To this end, we extend the popular web-based, open-source `bpmn-js`[1] process editor with elements and features targeted at BPMN choreography diagrams. We aim at

[1] `https://github.com/bpmn-io/bpmn-js`

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

a complete support of the BPMN choreography diagram standard, with novel features carefully derived from our own experience of using and modeling choreographies [1]. We already use `chor-js` successfully in our own projects and research, and hope to see others do so as well.

In this paper, we first give an overview of `chor-js` (see Sect. 2). We then discuss the maturity of `chor-js` in Sect. 3, in which we also introduce the sample use case of a diagram validator. Lastly, we provide the necessary information to access, use and contribute to `chor-js` in Sect. 4.

2 Overview & Features

`chor-js` is a web-based framework adapted to recent web browsers. Figure 1 shows the graphical user interface of our demo application. There are four principal components provided by the core `chor-js` library: (i) a left-hand side palette containing modeling tools and elements; (ii) a top palette providing a switching and renaming mechanism for managing multiple diagrams in one model; (iii) a context menu providing actions on the currently selected elements; and (iv) the editor itself filling the entire screen. Additionally, our demo application implements several extensions shown in a palette on the bottom left, exposing features like opening, saving or validating a diagram (v). Development and licensing information is given on the bottom right (vi).

In the scope of this paper, we can not highlight every feature provided by `chor-js`. Notable mentions include the many novel context actions tailored to specific modeling elements. For example, participant bands can be freely moved

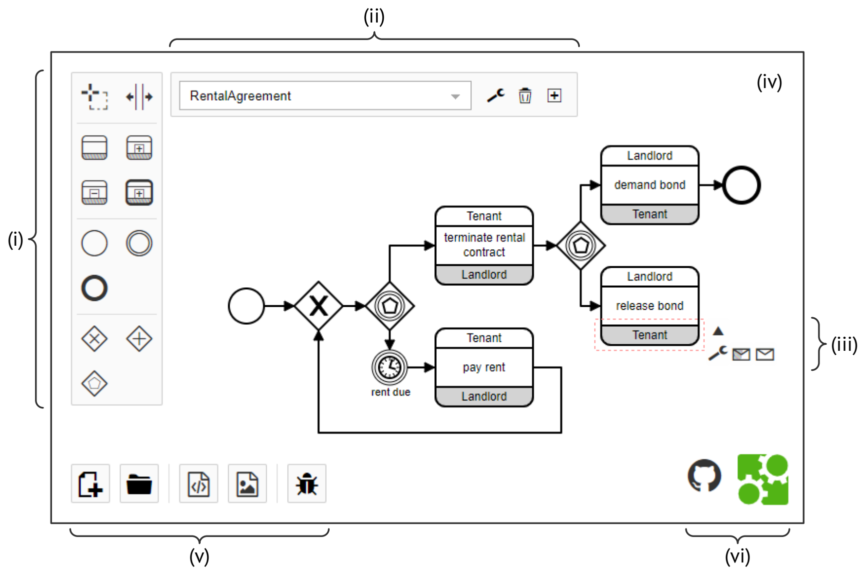


Fig. 1. Screenshot of the `chor-js` demo application

up and down; messages can be added, hidden or removed; and participants can be intuitively linked between call choreographies. Further, a prediction algorithm was devised which automatically populates new tasks with participants matching the context of the diagram, e.g., picking the correct initiator after merging several sequence flows. These features are mostly based on our own experience with other modeling tools, which are severely lacking in these functional aspects.

3 Maturity

While `chor-js` is still in active development, it is already reasonably stable and in use for several student projects. In this section, we will first shed light on the current level of standard coverage in comparison to other tools, and then provide an example for an extension towards a specific use case.

3.1 Standard Coverage & Tool Comparison

`chor-js` already provides support for most elements of the BPMN 2.0 choreography diagram standard, with some still being work-in-progress: Table 1 shows a non-exhaustive overview as well as a comparison with four different existing modeling tools, namely Signavio, Eclipse BPMN2 Modeler, Visual Paradigm and Trisotech. A green checkmark implies full support for a feature, a yellow bullet hints at partial or broken support, while a red cross signifies missing support.

For space reasons, we can not go into detail on all of the assessments. To give an example, we tested the tools' XML diagram interchange capabilities by importing official example models provided by the Object Management Group (OMG) [3]. Both Signavio and Visual Paradigm were not able to import these files properly, missing participant bands and attached messages, respectively. In general, tools would often ostensibly support a feature, but quickly exhibit

Table 1. Feature coverage of `chor-js` in comparison to other modeling tools

			chor-js	Signavio	BPMN2 Modeler	Visual Paradigm	Trisotech
features	code base		open	closed	open	closed	closed
	XML diagram interchange		✓	•	✓	•	✓
	standalone diagrams		✓	✓	✓	✓	✓
	between collaboration		×	×	×	•	•
element support	**gateways**		✓	✓	✓	✓	✓
	events	start, end, intermediate	✓	✓	✓	✓	✓
		boundary	×	•	×	×	•
	messages (attached)		✓	✓	✓	×	✓
	choreo. activities	choreo. task	✓	✓	✓	✓	✓
		sub-choreo.	✓	✓	•	•	✓
		call choreo.	✓	•	•	•	×
		global choreo. task	×	×	•	×	×
	markers	loop type	✓	✓	✓	✓	✓
		participant multiplicity	✓	×	✓	×	✓

problems when actually using it or exporting the model afterwards. For example, Visual Paradigm supports sub-choreographies, but does not display their contents correctly and exports erroneous diagram interchange information.

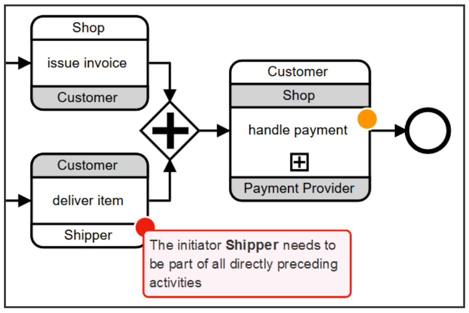
As is apparent from the comparison, `chor-js` provides the most complete choreography modeling experience to date — except for boundary events and global choreography tasks, all elements are supported. This has different reasons: Regarding the former, the metamodel defined in the BPMN standard fails to account for boundary events, making it impossible to connect them to choreography tasks [3]. Adding boundary events to `chor-js` would thus necessitate a metamodel extension, risking interchangeability with other tools. In fact, two tools (Signavio and Trisotech) allow modeling of boundary events in their modeling environment, but produce invalid XML when saving the diagram.

Global choreography tasks, on the other hand, are insufficiently specified in the standard, and their distinction from regular choreographies is unclear. During our extensive research regarding choreography diagrams, we have never encountered a single example of them in use. Additionally, `chor-js` does not yet support choreographies between collaborations, i.e., choreography diagrams between pools of a collaboration diagram.

3.2 Example Extension: Validator

One of the main contributions of `chor-js` is providing an extensible platform for projects using choreography diagrams as their conceptual basis in a modern, web-based fashion. To showcase these capabilities, we implemented a validator which provides basic validation capabilities for choreography diagrams based on the constraints posed in the BPMN 2.0 standard. The validator subscribes to the event bus of `chor-js` to receive updates and react accordingly.

Figure 2 shows an example of the validator feedback. The diagram models an ordering workflow which violates the simple flow constraint — the Shipper can not know when it is their turn to deliver the item since they were not part of the previous interactions. The error message hints at this, and users may resume to solve the issue, e.g., by adding and intermediate handover task.



The initiator **Shipper** needs to be part of all directly preceding activities

Fig. 2. Screenshot of the validator feedback on an invalid diagram

In its current version, the validator checks for errors in the sequence flow of choreography activities like in the example above [3, Sec. 11.5.6], incorrect usage of event-based gateways [3, Sec. 11.7.2] as well as correct call choreography linking. Additionally, a few style recommendations are provided via warnings. These include, for example, a warning if participants used inside a sub-choreography do not have a corresponding band — which is actually the warning that is visible on the sub-choreography in Figure 2.

4 Development & Resources

`chor-js` is implemented in JavaScript and has been tested with recent versions of the Mozilla Firefox, Google Chrome, and Apple Safari browsers. The project is maintained on GitHub and published through the *npm* package manager; see Table 2 for a collection of links. We further created a narrated screencast running through an example use case. A demo `chor-js-demo` showcasing the integration of `chor-js` in a Node.js application with additional features is provided as well. The demo application is updated regularly and deployed as a live version on our website. Instructions on how to set up `chor-js` and contribute to the project can be found in the respective repositories, among further documentation, tutorials and issue tracking that go beyond the scope of this paper.

Table 2. Locations of the projects and artifacts associated with `chor-js`

artifact	**location**
`chor-js` main application:	`https://github.com/bptlab/chor-js`
npm package:	`https://www.npmjs.com/package/chor-js`
narrated screencast:	`https://www.youtube.com/watch?v=x1bZlFXRusI`
`chor-js-demo` application:	`https://github.com/bptlab/chor-js-demo`
live version:	`https://bpt-lab.org/chor-js-demo/`

References

1. Ladleif, J., Weske, M., Weber, I.: Modeling and enforcing blockchain-based choreographies. (accepted at) Business Process Management (BPM) (2019)
2. Mendling, J., Weber, I., Aalst, W.V.D., et al.: Blockchains for business process management – challenges and opportunities. ACM Transactions on Management Information Systems (TMIS) **9**(1), 4:1–4:16 (Feb 2018), ISSN 2158-656X, https://doi.org/10.1145/3183367
3. OMG: Business Process Model and Notation (BPMN), Version 2.0.2 (Dec 2013), URL `http://www.omg.org/spec/BPMN/2.0.2/`
4. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: Business Process Management (BPM), LNCS, vol. 9850, pp. 329–347, Springer (2016)