fcm-js: A Tool for Design-Time Support of Fragment-Based Case Management

Kerstin Andree[1], Leon Bein[1], Maximilian König[1], Caterina Mandel[1], Marc Rosenau[1], Carla Terboven[1], Dorina Bano[2], Stephan Haarmann[2] and Mathias Weske[2]

Hasso Plattner Institute for Digital Engineering, University of Potsdam, Prof.-Dr.Helmert-Str. 2-3, 14482 Potsdam, Germany
[1]*{firstname.lastname}@student.hpi.de*
[2]*{firstname.lastname}@hpi.de*

Abstract
Fragment-based Case Management (fCM) is an approach to handle flexible, knowledge-intensive business processes. An fCM model is composed of four artifacts, modeling process behavior and data. Different modeling languages and hidden dependencies among these artifacts make modeling especially challenging. However, no adequate tooling that supports designers exists. To close this gap, we propose *fcm-js*, a modeling tool that provides a joint, visual user interface for all artifacts and integrates automated guideline checking based on fCM guidelines.

Keywords
Hybrid Process Modeling, Design-Time Support, Case Management, Process Modeling Guidelines

1. Introduction

Recently, knowledge-intensive processes receive more and more attention in the BPM community. Knowledge-work is highly flexible and driven by the decisions of experts (e.g., physicians). Existing BPM approaches provide insufficient support for these processes [1, 2, 3]. Thus, novel approaches have been proposed, one of which is *Fragment-based Case Management (fCM)* [4, 5]. fCM process models stand out because they consist of four artifacts: a set of process fragments, a domain model, a set of object lifecycles, and a goal state [4]. The use of different modeling languages in these artifacts and hidden dependencies among them make modeling challenging.

While tools for modeling with process-centered languages, such as *BPMN* [6], and with data-centered ones, such as *PHILharmonicFlows* [7], exist, hybrid approaches, such as fCM, lack up-to-date tooling that deals with their unique challenges and supports designers in creating high-quality models.

In this demo, we present the open-source fCM modeling tool *fcm-js*[1]. It enhances the modeling process by offering an intuitive user interface and improves the model quality by integrating several fCM modeling guidelines. *fcm-js* allows the user to model all artifacts of an fCM model at the same time, which gives a good overview of the progress and can speed up modeling. Furthermore, *fcm-js* includes an extensive guideline checking system. Currently, 20 guidelines regarding good fCM modeling[2] are implemented on different levels. Users are informed of

Proceedings of the Demonstration & Resources Track, Best BPM Dissertation Award, and Doctoral Consortium at BPM 2022 co-located with the 20th International Conference on Business Process Management, BPM 2022, Münster, Germany, September 11–16, 2022

[1]Repository with source code, documentation, and tutorial: https://github.com/bptlab/fCM-design-support
Exemplary deployment at: https://bpt-lab.org/fcm-js/
[2]https://github.com/bptlab/fCM-design-support/wiki

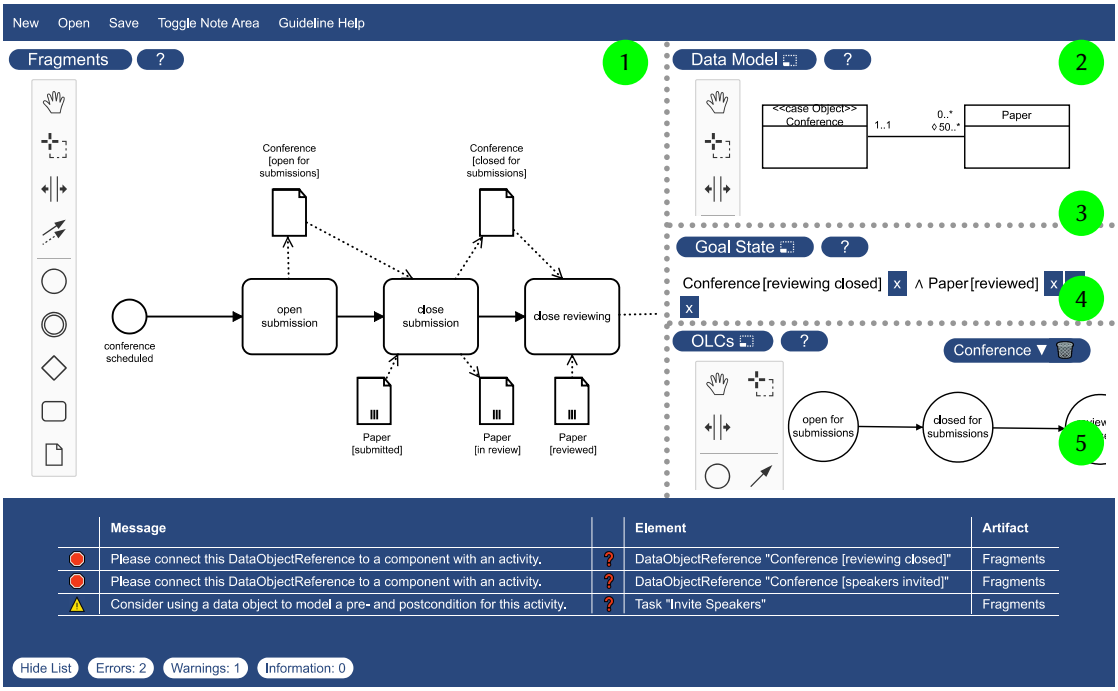

Figure 1: Screenshot of *fcm-js* user interface

potential errors and best practice violations in their artifacts, and they are offered quick-fixes. Moreover, some guidelines regarding the consistency between the artifacts are enforced by the tool, thereby ensuring a certain degree of quality. In the remainder, we present the main functionalities in more detail and discuss the maturity of the tool. For this, a small user study is presented, and future work is highlighted.

2. Overview and Features

fcm-js is a web application mainly written in HTML and JavaScript. One of its core features is that it is composed of one modeler for each artifact of fCM models. Figure 1 shows a screenshot with all important visual components: the modelers for (1) fragments, (2) data model, (3) goal state, and (4) object lifecycles (OLCs); (5) the guideline violation table. In the following, the modelers are described in more detail, and we briefly explain the implementation of two other core features, namely consistency and guideline checking.

2.1. Individual Artifact Modelers

The individual modelers are technologically based on *diagram-js*, a "toolbox for displaying and modifying diagrams on the web"[3]. Hence, they are designed similarly: as user interface, SVG-based diagram canvases are provided, featuring drag-and-drop-based visual modeling

[3]https://github.com/bpmn-io/diagram-js

and advanced features such as copy and paste. As interface for custom internal modules and auxiliary components, each modeler features an event bus, which can be used to listen to, handle, and fire various events. It is the key component to communicate with the modelers on a software level.

The *Fragment Modeler* builds upon the BPMN modeler *bpmn-js*[4], as fCM fragment models use a subset of BPMN. The Fragment Modeler supports users, for example, when placing data object references by making them choose data classes and states from the data and object lifecycle models. The *Data Modeler* allows creating basic UML class diagrams consisting of classes with names and attributes, and associations with multiplicities and fCM-specific goal multiplicities. In the *Goal State Modeler*, states from the OLCs can be put together in disjunctive normal form with dropdown menus. The *Object Lifecycle Modeler* allows selecting a data class whose OLC should be displayed. OLCs are modeled exclusively with states and unlabeled transitions.

2.2. Modeler Interplay

One inherent challenge of fCM modeling is having to work in four models at the same time, and ensuring their consistency with each other.

fcm-js displays all four modelers side by side (cp. Fig. 1). It allows users to adjust the spacing on-the-fly and select which modeler to display "in focus" on the left side. This way, users can seamlessly work on multiple artifacts at the same time, avoiding mental context switches.

Furthermore, *fcm-js* enforces general consistency between the artifacts. To this end, we put a *mediator* component in place, which connects to each event bus. For instance, deleting and renaming data classes or OLC states is captured and propagated across the modelers, which can then adapt their models automatically.

All artifacts can be imported and exported as one file, allowing users to save their models and continue modeling later on.

2.3. Guideline Checking

Modeling guidelines help designers to detect errors and to assess the quality of a model. However, checking guidelines manually is tedious and error-prone. With the mediator component, *fcm-js* already automatically enforces most consistency-based guidelines from the fCM guideline catalog[5] . For many other guidelines, *fcm-js* features automated checking. Guidelines are translated into one or more checks, which are unambiguously met or violated by a given model. These checks are reevaluated whenever the user changes the model.

As an example, consider the guideline with catalog ID C3[6], which states that state transitions induced by activities should be covered by the OLCs. Furthermore, consider the process fragment displayed in Fig. 2, which shows the activity *Invite Speakers* that changes instances of class *Conference* from state *reviewing closed* to state *speakers invited*. If there is no matching transition in the OLC of class *Conference*, this fragment violates C3. In *fcm-js*, this violation is detected, and the user is made aware of it by highlighting the activity (cp. Fig. 2) and listing the violation

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

[5]https://github.com/bptlab/fCM-design-support/wiki

[6]https://github.com/bptlab/fCM-design-support/wiki/Consistency#c3

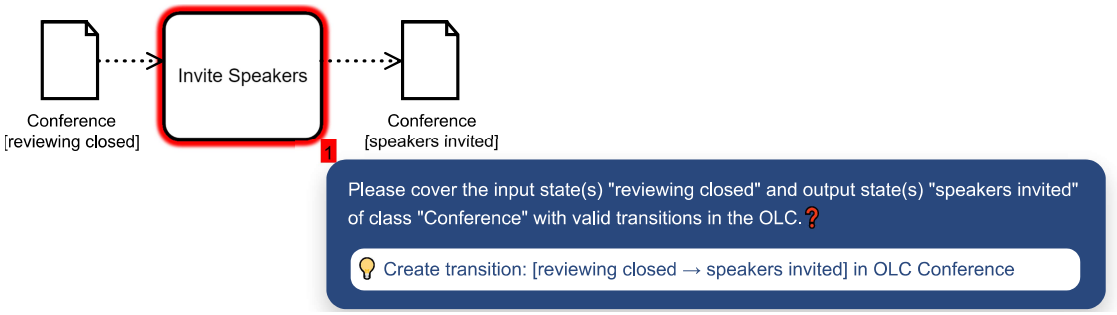

Figure 2: Violation highlighting, a clipping from the fragment modeler of *fcm-js*

in a hideable guideline violation table (cp. (5) in Fig. 1). For each violation, *fcm-js* provides an explanatory message, and optionally a list of quick-fixes, i.e., concrete actions to fix the violation that can be applied automatically. In the example, clicking on the first option would automatically create the missing OLC transition for the user, which solves the violation.

Immediate feedback through automated violation detection and highlighting helps designers to learn fCM in a trial-and-error fashion. Furthermore, quick-fixes accelerate the user workflow by resolving violations more quickly.

3. Maturity

While *fcm-js* is still active in development, it is already in a stable version and can be used in projects. In this section, we will first showcase two small user studies conducted with the tool and then provide an outlook for future work.

3.1. User Studies

We assessed the effectiveness of *fcm-js* though a user study comparing the quality of fCM models created by hand with models created with *fcm-js*. Two groups of each five competent fCM designers modeled the same knowledge-intensive process. One group received the guideline catalog and *fcm-js*, the other did not. Manually checking created models showed an increase of fulfilled guidelines in the tool-supported group (66% vs. 84%). This effect was even higher when only considering guidelines that were implemented in *fcm-js* (66% vs. 91%). With that, we conclude that *fcm-js* helps designers to fulfill guidelines and thus produces fCM models of higher quality.

Furthermore, we conducted think-aloud sessions and interviews with five different competent fCM designers, who were asked to model an fCM case model based on a textual description using *fcm-js*. The participants stated that they felt more confident in modeling and rated their models as higher in quality compared to previous modeling procedures. In particular, the general tool structure and automatically performed consistency checks were positively highlighted. We infer that *fcm-js* improves the fCM modeling experience.

3.2. Future Work

Our tool is a stand-alone web application that focuses on the design-time support and aims to make fCM more accessible to novice users. Future work could consider an integration into existing fCM execution pipelines. Regarding the expandability of *fcm-js*, more guidelines can be easily added through the unified guideline interface.

Furthermore, user experience can be improved. For instance, undoing and redoing are currently not possible in *fcm-js* and could be a significant enhancement. However, especially considering our strict consistency guidelines which automatically synchronize all artifacts, implementing un- and redoing is a challenging task.

In general, the design of *fcm-js* can be transferred to future projects on hybrid modeling approaches. We want to stress especially the side-by-side layout and the consistency and guideline checking, which helped designers to cope with the complexity induced by composed models.

References

[1] C. D. Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches, J. Data Semant. 4 (2015) 29–57. doi:10.1007/s13740-014-0038-4.

[2] K. D. Swenson, Position: BPMN is incompatible with ACM, in: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers, 2012, pp. 55–58. doi:10.1007/978-3-642-36285-9_7.

[3] W. M. P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Comput. Sci. Res. Dev. 23 (2009) 99–113. doi:10.1007/s00450-009-0057-9.

[4] M. Hewelt, M. Weske, A hybrid approach for flexible case modeling and execution, in: Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings, volume 260 of *Lecture Notes in Business Information Processing*, Springer, 2016, pp. 38–54. doi:10.1007/978-3-319-45468-9_3.

[5] S. Haarmann, Fragment-based case management models: Metamodel, consistency, & correctness, in: Proceedings of the 13th European Workshop on Services and their Composition (ZEUS 2021), Bamberg, Germany, February 25-26, 2021, volume 2839 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 1–8. URL: http://ceur-ws.org/Vol-2839/paper1.pdf.

[6] F. Corradini, A. Ferrari, F. Fornari, S. Gnesi, A. Polini, B. Re, G. O. Spagnolo, A guidelines framework for understandable BPMN models, Data Knowl. Eng. 113 (2018) 129–154. doi:10.1016/j.datak.2017.11.003.

[7] S. Steinau, K. Andrews, M. Reichert, A modeling tool for philharmonicflows objects and lifecycle processes, in: Proceedings of the BPM Demo Track and BPM Dissertation Award (BPM 2017), Barcelona, Spain, September 13, 2017, volume 1920 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1920/BPM_2017_paper_196.pdf.