Dynamic Evaluation Forms using Declarative Modeling

Rasmus Strømsted[1], Hugo A. López[2,3], Søren Debois[2], and Morten Marquard[3]

[1] Exformatics A/S
ris@exformatics.com
[2] IT University of Copenhagen, Denmark
{hual,debois}@itu.dk
[3] DCR Solutions A/S
{hual,mm}@dcrgraphs.net

Abstract. This paper reports preliminary experiences using the Dynamic Condition Response (DCR) graphs declarative process notation to specify dynamic evaluation forms, and using an execution engine for that notation to subsequently "run" the form. The DCR notation was able to express *all* the patterns of behaviour necessary for a real case: a post-hoc evaluation form for a Danish arbitration court, "Voldgiftsnævnet for Anlæg og Byggeri". However, some patterns were somewhat cumbersome to express.

Keywords: forms, declarative process model, Dynamic Condition Response, DCR, process execution

1 Introduction

This paper presents a preliminary experience report using the DCR Solutions declarative process engine [4, 3, 2] as a mechanism for implementing dynamic web forms [7] in Exformatics ECM. We investigate the question whether the Dynamic Condition Response (DCR) graph process language is flexible enough to model the requirements of a real-life evaluation form example, by applying the methodology to an evaluation form commissioned by the Danish arbitration court "Voldgiftsnævnet for Anlæg og Byggeri" (VBA).

Related work Forms have strong similarities with declarative process notations explored [7]: (1) The order of execution is flexible: the user may fill in the form in whatever order he prefers, and (2) forms have constraining rules, e.g. "*If field A is filled in, then also field B must be filled in*".

The premiere extant commercially/freely available services for constructing on-line evaluation forms are provided by SurveyMonkey and Google Forms. Both products emphasize ease-of-use, both for the form constructor and the end-user.

A SurveyMonkey form is specified as a list of fields. Each field has a type (e.g., multiple choice, number, text, etc.), and the form designer may choose the ordering of fields. Both SurveyMonkey and Google Forms offer dynamically

F. Casati et al. (Eds.): Proceedings of the Dissertation Award and Demonstration, Industrial Track at BPM 2018, CEUR-WS.org, 2018.

changing behaviour: 1. SurveyMonkey has so-called "skip page logic". This feature allows the form designer to set conditions to control the sequence flow of the form. In SurveyMonkey this is done by grouping questions in pages, and using conditional statements on questions to control whether or not the group of questions will be shown. 2. Randomized sequence features, which shuffles the evaluation question 3. Dynamic labeling, which SurveyMonkey calls "Question & Answer Piping". This feature allows the form designer use text variables in question labels; the value of the variable is set to the answer of a given previously answered question.

2 DCR Graphs

In this section, we give a brief introduction to the Dynamic Condition Response (DCR) graph process notation [1, 5]. A DCR graph is a directed multigraph where nodes are activities, and edges are relations between activities. Each activity has an associated state (i.e.: Executed, Included, and Pending). The "Executed" state indicates whether the activity was previously executed. The "Included" state indicates whether it is currently relevant for the workflow. Finally, the "Pending" state indicates whether the activity must execute (or be made not "Included") before the workflow can be considered complete.

DCR is a declarative notation: processes are specified not by explicit sequence flows, but by the *rules* (relations) governing which sequence flows are admitted and which are not. Relations between activities indicate (1) constraints on whether an activity may be executed depending on the states of other activities, and (2) how executing one activity may affect the state of another.

E.g., suppose (a) that an activity "Payout" is constrained to not happen unless "Approve payout" happened first; and (b) that executing "Payout" changes the state of itself to be "no longer relevant" (to prevent a second payout).

DCR Solutions provides an on-line process portal[4], for creating and simulating DCR graphs [3]. DCR Graphs answer to the following graphical notation:

- Activities: First class citizens, they are denoted by grey boxes.
- Condition (Yellow arrow): A constraining relation, e.g., activity A has to be executed before activity B.
- Response (Blue arrow): A follow-up relation, e.g., after activity A has been executed activity B is set to 'Pending', which means that B must happen or be excluded for the workflow to complete.
- Exclude (Red arrow): Makes an activity irrelevant, e.g., if activity A is executed, activity B is set to 'not included'. Irrelevant activities are disregarded as conditions and cannot be executed.
- Include (Green arrow): Opposite of Exclude: re-instates an activity.

[4] `http://dcrgraphs.net`

3 DCR Forms

In order to use DCR Graphs to build a form, we interpret activities as fields in the form, then we use relations to define the dynamic behaviour, e.g., to turn on and off one form field/activity depending on the value of another field.

Fig. 1 provides an example of a form in DCR graphs. There are six activities. The title of the activity corresponds to a field text question in the form. An activity is executed with a green tick, and solid (resp. dashed) borders in an activity denote whether it is included (resp. excluded). A pending activity has a blue exclamation mark. Colored arrows denote relations in Sec. 2.

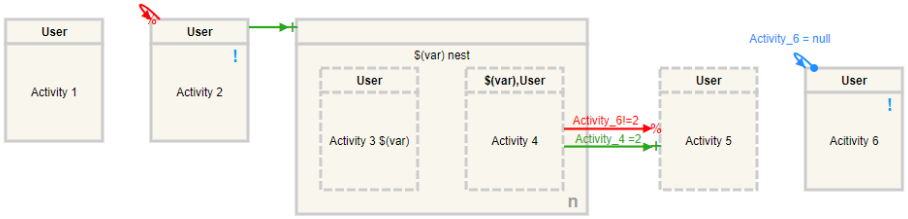

Fig. 1. Example of an evaluation form using DCR Forms.

Notice how states of activities are drawn in Fig. 1, with three activities initially excluded. Two activities have pending states, indicating that they must be executed or excluded in the form to be deemed complete.

Each activity has a role and a label onto it. The green relation arrow from "Activity 2" includes a parent activity, including the nested activities "Activity 3 $(var)" and "Activity 4". Relations may have guards, as exhibited in the arrows between "Activity 4" and "Activity 5". They are used to constrain behaviour based on input.

Fig. 1 uses some modelling techniques for creating forms using DCR graphs:

1. **Mandatory fields as pending activities**: The pending state of "Activity 2" indicates that it must be executed before the workflow is complete. For forms, we interpret this as the field "Activity 2" being mandatory: The form cannot be submitted unless "Activity 2" has received a value.
2. **Guarded inclusions/exclusions**: Dual, guarded include/exclude arrows from "Activity 4" to "Activity 5" represent a common pattern that includes or excludes "Activity 5" in the workflow/form depending on the value of "Activity 4".
3. **Self-responses as content-control**: A self-response on "Activity 6" is guarded by "Activity_6 = null". Since "Activity 6" is already a mandatory field because of the blue exclamation mark, it ensures that if the field is filled, and it is cleared at a later stage, it goes back to a pending/mandatory state.

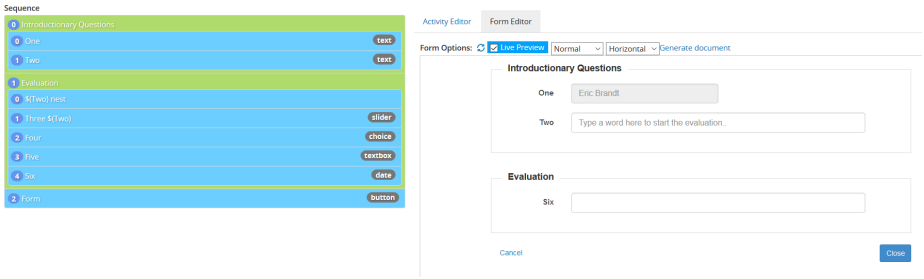

Fig. 2. The sequence editor (left) and form previews (right).

The following new features in the DCR process portal were useful for the development of declarative forms:

Sequence Editor and Form Preview The DCR process portal allows setting sequencing and grouping fields by drag-and-drop order controls. Fig. 2 shows on the left side the order of the sequence editor tool. When the form is shown in the preview (right side), it simulates what the recipient will see. Notice that only the included activities of Fig. 1 are shown.

Dynamic Labels. DCR process portal supports dynamic labels. In Fig. 1 the activity "Activity 3" has label "Activity 3 $(var)". At run-time this label will evaluate the contents of variable "var". This variable is bound to the input given at "Activity 2". The variable can be used for both roles and activity labels.

4 Evaluation Form for Voldgiftsnævnet's Arbitration Proceedings

We now report on using the above mechanisms to specify a real-life for the VBA case. The initial requirements for VBA's evaluation form were simple: After an arbitration, VBA would send an evaluation form to the involved parties. The evaluation questions included how satisfied parties were with VBA as arbitrators, with the process followed, and with the outcome of the proceeding. The exact questions were outlined by VBA, and Exformatics produced from this specification a DCR graph to be interpreted as a web form.

The VBA specification actually called for evaluation of five different types of arbitration proceedings, called (A), (G), (P), (Q), (C). An evaluation comprised both a set of general questions and a set of type-specific questions. The (C) type was a special case, and had choices which would include other proceeding types. Some of the proceeding types shared specific questions e.g. (P) and (G).

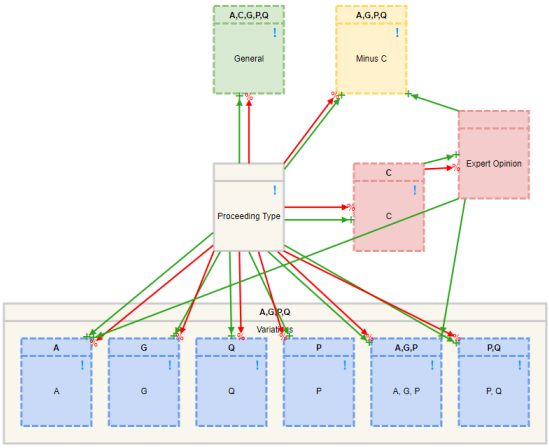

Fig. 3. DCR Graph of the VBA Evaluation Form structure.

4.1 VBAs Evaluation DCR Graph

The structure of the model is given in Fig. 3[5]. Exformatics used roles to denote the arbitration proceeding type, and colors to ease the readability of the graphical presentation. The choice of which evaluation form to present was made using the guarded include/exclude pattern: A choice field allowed the user to pick a type of proceedings, who will control the remaining content to be shown.

Exformatics and VBA filled the evaluation structure with the questions outlined, see Fig. 4. The model uses guards to impose the desired behaviour; e.g.: activity C will be only included if the case type ("Sagstype" in Danish) required is "C", and will be excluded otherwise.

Robotic support: Activities in Fig. 4 that correspond to the 'Exformatics' role are executed automatically. The data required for such activities is collected from the VBA database. The data was used to set variables for use as dynamic labels.

4.2 Adaptation

VBA intended to continuously improve the form *after* its initial development, and they needed to collect statistics on the answers collected. It was therefore vital that (1) the DCR graph could be accessed and modified by VBA staff; (2) that VBA staff could in fact understand and modify the graph; and (3) that VBA could have a data-extraction process that was stable under graph updates.

A key challenge was that VBA staff on the project *did not* have prior programming experience. However, the staff quickly acquired basic modelling competences, and contributed to model development. Nonetheless, understanding elements of the DCR Graph remained cumbersome for some staff.

[5] The full graph is accessible at `http://www.dcrgraphs.net/tool/main/Graph?id=da7c4489-578c-4b94-b17c-66ceb214692c`

5 Discussion

Unlike the previous use cases for DCR as declarative processes [7], evaluation forms require further data field support. For instance, to express questions such as "On a scale from 1 to 10, how much do you like...". A 'slider' field type was added to the DCR Form, which lets the user drag a handle to on a scale set by the form designer. In future work we would like to include further support for additional field types, e.g.: Multi-checkbox, Star rating, Matrix scale, Image choice, etc. This is, however, orthogonal to the behaviour of the form.

Mapping some standard behaviour of forms into DCR models proved to be challenging. That was the case of mandatory fields: One must add a pending activity combined with a self-response guarded by null, as discussed above. As seen in Fig. 4 *most* activities use this technique. In further work, we will consider adding process templates to the DCR portal to encapsulate this kind of behaviour, especially for readability.

In the forms case, include/exclude relations tend to come in pairs corresponding to the true and false branches of a condition. It would be helpful to express such choices in a *single* relation. The image below (Fig. 5) illustrates how the VBA evaluation graph might look had such a relation been available.

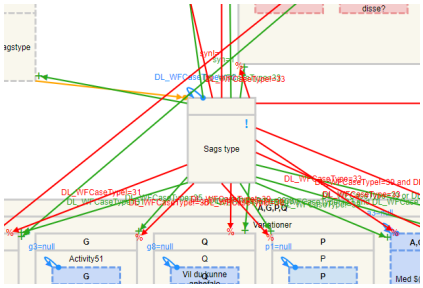

Fig. 4. Excerpt of actual model. Note the pervasive green-red inclusion-exclusion examples and blue self-responses.

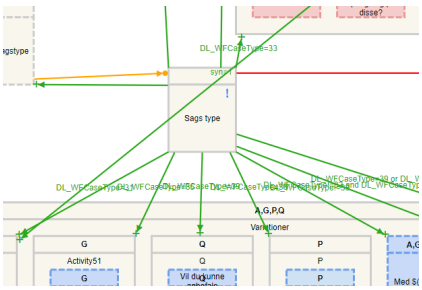

Fig. 5. Excerpt of hypothetically improved model.

One of the main differences between DCR and SurveyMonkey is the control of the sequence flow. Compared to DCR, SurveyMonkey offers only a limited version of declarative conditional rules. SurveyMonkey only supports one condition for each question and SurveyMonkeys "skip" logic is bound to page breaks, in contrast to DCR which responds with change every time the user engage with a field. The flexibility of DCR was in fact instrumental to satisfy the requirements made by the customer. For instance, without the inclusion of DCR relations, it would have been challenging in SurveyMonkey to model VBAs evaluation, whereas this comes more naturally in DCR.

The second difference was economy: by using DCR Graphs's declarative notation we were able to minimize the number of fields produced, enabling them depending on the type of proceeding. As a consequence (1) From a maintenance perspective, new changes to the forms have become agile; change is done to one activity, not to several. (2) From a data analysis perspective, the data from the different types were directly compatible on extraction, not having to merge data from the different proceeding types.

As mentioned in Sec. 4.2, one of the challenges remaining is to bridge the adoption gap between declarative models and users with no experience in declarative modelling, for instance, lawyers. In future iterations we would like to explore how tools that maintain the correspondences between natural language specifications and DCR graphs (i.e.: [6]) help us bridging such a gap.

6 Conclusion

The DCR graphs formalism successfully expressed the required dynamic behaviour of the VBA evaluation form. However, the modelling was not entirely straightforward, as the addition of Dynamic Labels and new a field type was introduced, and expressing dynamic behaviour via DCR relations remain in some cases artificial. In the future, DCR Solutions will study the inclusion process templates in order to facilitate abstraction and process reuse in the tool.

Acknowledgments Work supported by the Innovation Fund Denmark project EcoKnow.org (7050-00034A). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement BehAPI No.778233.

References

1. Debois, S., Hildebrandt, T.: The DCR Workbench: Declarative Choreographies for Collaborative Processes. In: Behavioural Types: from Theory to Tools, pp. 99–124. River Publishers (Jun 2017)
2. Debois, S., Hildebrandt, T., Marquard, M., Slaats, T.: Hybrid Process Technologies in the Financial Sector: The Case of BRFkredit. In: Business Process Management Cases, pp. 397–412. Management for Professionals, Springer, Cham (2017)
3. Debois, S., Hildebrandt, T.T., Marquard, M., Slaats, T.: The DCR Graphs Process Portal. In: BPM Demo Track 2016. pp. 7–11 (2016)
4. Debois, S., Hildebrandt, T.T., Slaats, T., Marquard, M.: A Case for Declarative Process Modelling: Agile Development of a Grant Application System. In: EDOC '14. pp. 126–133. IEEE Computer Society (2014)
5. Hildebrandt, T., Mukkamala, R.R.: Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs. In: Post-proceedings of PLACES 2010. EPTCS, vol. 69, pp. 59–73 (2010)
6. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: From texts to declarative processes and back. In: BPM Demo Track 2018 (2018)
7. Marquard, M., Debois, S., Slaats, T., Hildebrandt, T.: Forms are declarative processes! (2016), presented at the BPM 2016 Industry track