DECLARE Demo: A Constraint-based Workflow Management System

Maja Pesic, Helen M. Schonenberg, and Wil van der Aalst

Department of Mathematics and Computer Science,
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, The Netherlands
m.pesic,m.h.schonenberg,w.m.p.v.d.aalst@tue.nl

Abstract. Mainstream workflow management systems are using procedural languages ranging from BPMN and EPCs to BPEL and YAWL. By demonstrating DECLARE, we will show that it is also possible to use a fundamentally different approach based on constraints. DECLARE allows for multiple constraint-based languages whose semantics are grounded in Linear Temporal Logic (LTL). The DECLARE system provides a broad range of functionalities ranging from design, enactment and dynamic change to verification, discovery and recommendation. This demo presents the main functionalities of DECLARE and is intended for both researchers and practitioners interested in innovative BPM solutions for processes that require flexibility.

1 Introduction

DECLARE[1][1, 2] is a constraint-based WFMS. The core of the system consists of three components, as shown in Figure 1. First, the Designer component is used for creating constraint templates, defining organizational structures, creating and verifying constraint models. Second, instances of constraint models are enacted and dynamically changed in the Framework tool. Finally, each user

[1] http://declare.sf.net

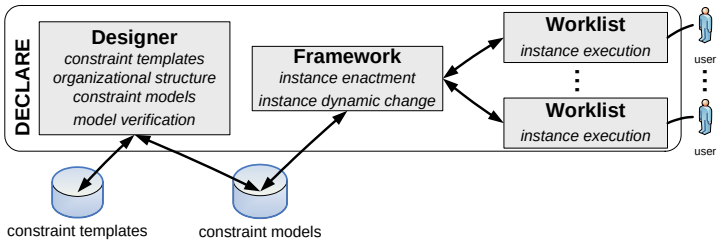


Fig. 1. Architecture of DECLARE

uses his/her Worklist component to access active instances and execute their tasks. DECLARE is a Java desktop application and is distributed under terms of the GNU General Public License. There is also a tight coupling between DECLARE and ProM[2] allowing for innovative forms of analysis and support, e.g., recommending particular process paths based on historic information.

DECLARE is significant and innovative in the BPM field because it uses a declarative constraint-based approach instead of the procedural one. In particular, DECLARE illustrates how declarative approaches can indeed be used to realize more flexible BPM solutions, while providing for various types of support [1]. However, the constraint-based approach (and DECLARE) is suitable for smaller business processes: using this approach for complex processes significantly reduces the efficiency and usability of the tool .

2 Main Functionalities

This demo presents the five main functionalities of DECLARE.

First, constraint templates are created on the system level as types of constraints. Figure 2 shows that a template is graphically represented as a special line between tasks. The formal specification of the template is given as a Linear Temporal Logic formula.

[2] http://prom.win.tue.nl/research/wiki/

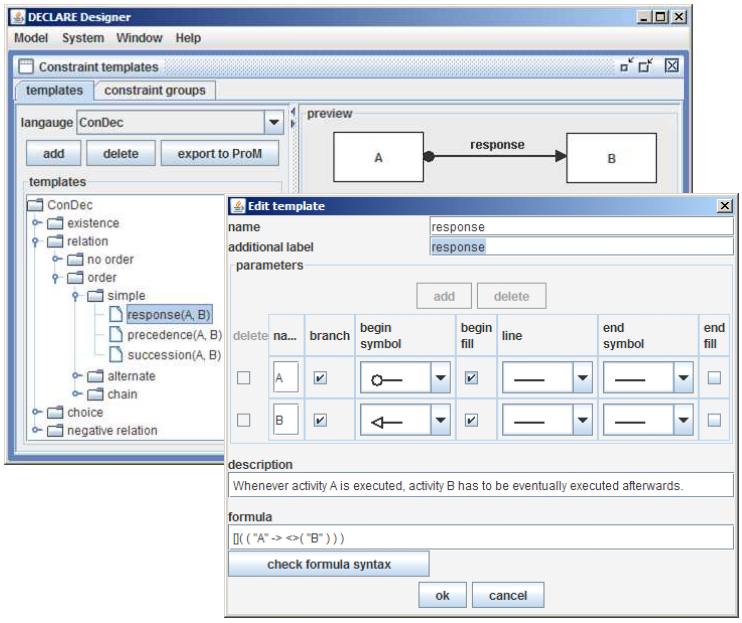
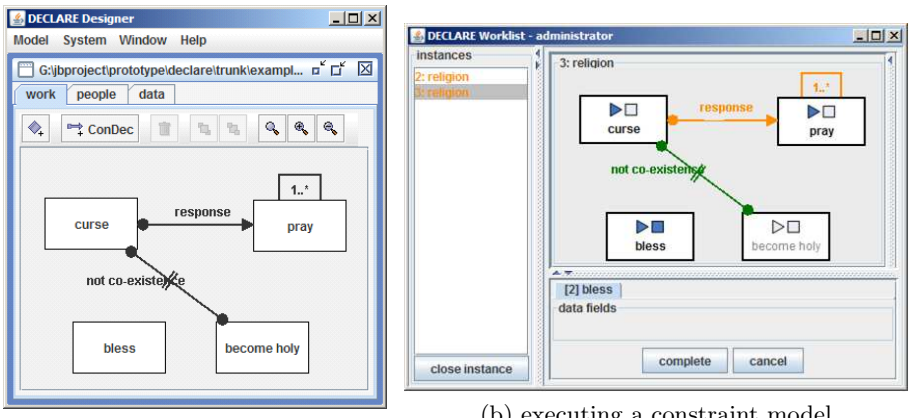


Fig. 2. Creating the *response* template

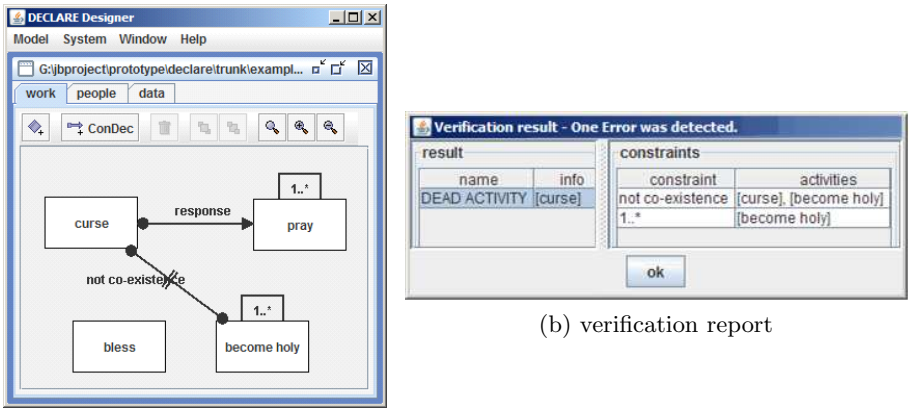


(a) creating a constraint model

(b) executing a constraint model

Fig. 3. Creating and executing constraint models

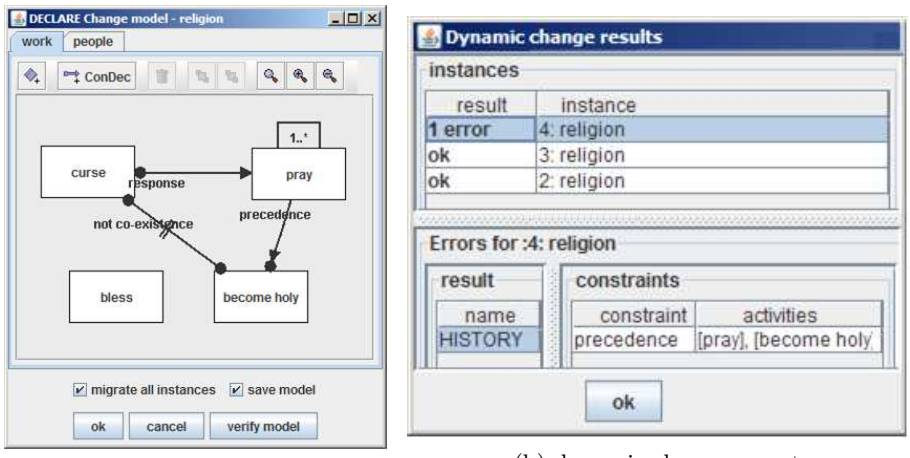
Second, constraint models are created by adding tasks and using constraint templates to create constraints between tasks. Figure 3(a) shows a model with tasks *curse*, *pray*, *bless* and *become holy*, and two constraints. Constraint *response* specifies that every time one *curses*, one has to eventually *pray* afterwards. Constraint *1..** specifies that one has to *pray* at least once. Third, Figure 3(b) shows how a model from Figure 3(a) is executed in the Worklist. The whole model is shown to the user, tasks are executed by double-clicking, and states of constraints are presented via special colors: red for *satisfied*, orange for *temporarily violated* (*not* satisfied at the moment, but *can* become satisfied in the future) and red for *permanently violated* (*not* satisfied at the moment, and *cannot* become satisfied in the future).



(a) a model with an error

(b) verification report

Fig. 4. Verifying constraint models



(a) adding a constraint dynamically

(b) dynamic change report

Fig. 5. Dynamically changing instances of constraint models

Fourth, Figure 4(b) shows how our verification procedure detects an error in the model model shown in Figure 4(a): task *curse* is dead. This is due to constraints *not co-existence* between tasks *curse* and *become holy*, which specifies that one cannot both *curse* and *become holy*, and *1..** on task *become holy*, which specifies that one must *become holy* at lest once.

Finally, Figure 5(a) illustrates the support for dynamic changes in DECLARE. Constraint *precedence* between tasks *pray* and *become holy* is added to all current and future instance of the model. Because this constraint requires that one must *pray* before one *becomes holy*, and task *become holy* has already been executed before task *pray* in instance number 4, this change is applied to all active instances except the instance number 4 (cf. Figure 5(b)).

3 Conclusions

DECLARE is a fully functional constraint-based WFMS, and it allows for creating, verifying, executing and dynamically changing constraint-based process models. DECLARE proves that a declarative approach can be applied to WFM, which makes WFMSs more flexible.

References

1. W.M.P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development*, 23(2):99–113, May 2009.
2. M. Pesic, M.H. Schonenberg, and W.M.P. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, pages 287–298, Washington, DC, USA, 2007. IEEE Computer Society.