The bflow* Hive – Adding Functionality to Eclipse-based Modelling Tools

Ralf Laue[1], Arian Storch[2], and Felix Höß[3]

[1] University of Applied Sciences of Zwickau, Department of Information Science
Dr.-Friedrichs-Ring 2a, 08056 Zwickau, Germany
ralf.laue@fh-zwickau.de
[2] it factum GmbH
Hainstr. 6, 04109 Leipzig
arian.storch@it-factum.de
[3] University of Hamburg, Chair of Information Systems, Germany
Felix.Hoess@studium.uni-hamburg.de

Abstract. The *bflow* Hive* is a collection of Eclipse plug-ins that can be integrated into various Eclipse-based modeling tools. It adds several features that are useful in particular for model analysis and model exchange. Most of these features put the principle of end user-programming into practice: Modeling tool users can add their own add-ons at runtime without having to re-compile the original modeling tool. We hope that this will be useful in particular for researchers. They can extend a modeling tool with own model analysis features without having to touch the sources of the modeling tool.

1 Introduction

The *bflow* Toolbox* [1] is an *Eclipse*-based open-source tool for graphical business process modeling using the Event-Driven Process Chains notation. From the beginning, it was the aim of its developers to make the tool open for contributions. While this is quite normal for an open-source product, we tried to go one step further. We provided interfaces to third-party tools such that those tools can be integrated into our modeling tool without having to become familiar with our source code or with *Eclipse* programming. A survey by Hannebauer et al. [2] showed that a major part of the barrier to contribute to open-source software is the effort invested until the first build. By lowering this barrier, we hope that more contributors can be encouraged to add their own extensions to our modeling tool.

While in the first years, our work concentrated on our own tool, we soon realized that this idea could be useful for other graphical modeling tools as well. Therefore we refactored our code which resulted in Eclipse plug-ins that can

be used with other modelling tools based on Eclipse using EMF and GMF or Graphiti. As a result, modellers and researchers can add their own extensions to those tools.

2 Features

2.1 Export / Import

We provide two ways for adding a transformation into or from an external file format: either by using XSLT or by using a transformation language which is based on the template language *Apache Velocity*. A new export/import can be added by creating a transformation script file, another file defining the transformation metadata (for instance, name or description) and placing both into a subdirectory of the modelling tool's workspace.

2.2 User-Defined Attributes and Icons

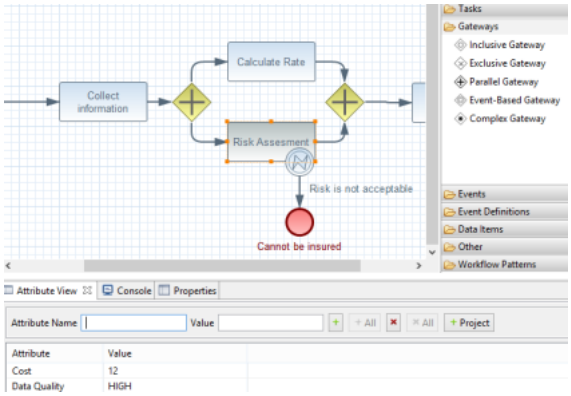
For model analysis purposes, it is often desirable to add custom-defined attributes to model elements (for example information about cost and duration to an activity or a probability to a control-flow edge). Unfortunately, this is not always supported by the original modelling tool. We created a new Eclipse view called "Attribute View" which allows to add arbitrary attributes to model elements. An attribute can be a numerical or textual value or an URL. In the latter case, the resource located by the URL can be opened from the Attribute View. This makes it possible to attach files or web sites with additional information to any node or vertex in the model. Additionally, a "Filter View" allows to show only those model elements whose attributes have a certain value (for example all activities where the attribute "cost" is larger than a given value).

It is possible to define that the shapes of the diagram should be decorated with icons when the value of a user-defined attribute is within a given range. For this purpose, the icon graphics and a map from attribute values to icons to be shown has to be added to a subfolder of the modeling tool. Using this mechanism, we were able to include the EPC model variant for Risk-aware Event-Driven Process Chains [3] into the *bflow* Toolbox* without the need to change the metamodel. Fig. 1(a) shows the Attribute View integrated into the *BPMN2 Modeler* (www.eclipse.org/bpmn2-modeler/); Fig. shows 1(b) an event in the *bflow* Toolbox* with a decorating "Risk" icon.

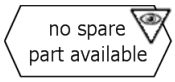
2.3 Model Validation

For improving the model quality, it can be desirable to give the modeller feedback about possible modelling problems or detected anti-patterns at modelling time.[1] For this purpose, we provide a mechanism to define checking rules in three

[1] The reader who is interested in this, might alternatively consider to use the *EMF Refactor* project, see www.eclipse.org/emf-refactor [4]



(a) User-Defined Attributes added to the *BPMN2 Modeler*



(b) Shape with Icon

Fig. 1. Adding Attributes and Icons

languages: the Epsilon Validation language (EVL), openArchitectureWare Check and Prolog. For rules expressed in EVL, it is also possible to define so-called "Quick-Fixes", i.e. model transformations that correct a problem.

The *bflow* Hive* provides the functionality for adding user-defined validation rules, for selecting the rules that should be applied, for the internationalisation of problem messages and for changing the problem messages according to the users' preferences. Fig. 2 shows how validation rules have been integrated into *openOME*, a tool for goal-oriented requirements modelling.

2.4 Add-ons

The framework allows to define "add-ons". An add-on can be described as a sequence of steps which defines how to...

1. Extract the model data and translate it into the input format that is required by an external program.

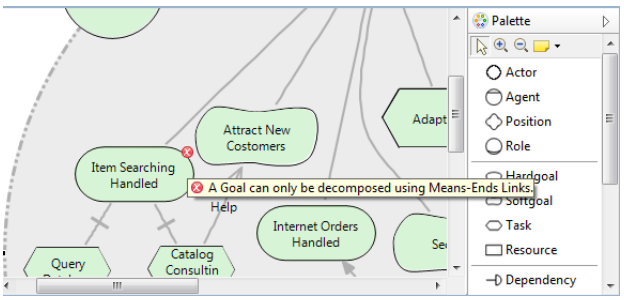


Fig. 2. Validation in *openOME*

2. Call the external program or a chain of such programs with specified parameters.
3. Read the result data of the program(s).
4. Show the result by changing one or more views in the modeling tool.

For each of those steps, our framework already provides out-of-the-box components which just need to be invoked in the right order and with the right parameters.

Fig. 3 shows possible executions of an add-on. As illustrated, the components are combined using a pipes-and-filters architecture.

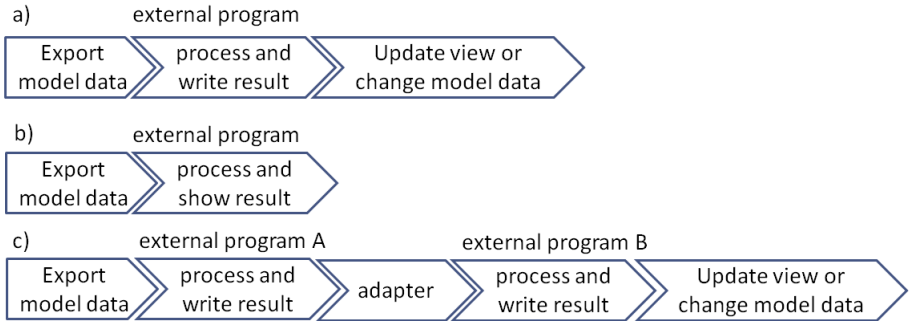


Fig. 3. Components of an Add-on

The called program(s) deliver the result by writing either into a file or to the standard output stream of the process. To update a view of the model within the modeling tool, the rightmost component in Fig. 3 a) requires an input in form of predefined commands. If the external program does not provide the output in the required format, we have to call another program as an adapter. For showing the results of the external program(s), it is possible to use components that can do the following actions:

- Printing information into the Eclipse Console View
- Adding information about an error, warning or information Eclipse's *Problems View*
- Changing the label or the graphical attributes (e.g. size, color, font-style) of model elements
- Changing the user-defined attributes of model elements (in addition, this can result in decorating a model element with icons) (as shown in Fig. 1(b))
- Showing a generated HTML page

With the help of the add-on mechanism, extensions in great variety have been developed. There are add-ons for model-checking, quality-of-service calculation for business processes, business process simulation [5], risk management, simulation of projects expressed as IDEF-0-like diagrams [6], checking the conformance of element labels to grammatical style rules and for finding modeling anti-patterns

[7]. Thanks to the add-on framework in the *bflow* Hive*, it was possible to add all this functionality without having to change the sources of the modeling tool.

3 How to Obtain

The *bflow* Hive* is already integrated into our own EPC modelling tool *bflow* Toolbox* (http://bflow.org), into UPROM [8] (a modeler that allows functional software size estimation) and into the requirements modeling tool openOME (http://sourceforge.net/projects/openome/).

The source code can be obtained from https://github.com/bflowtoolbox/app where all plug-ins named org.bflow.toolbox.hive.* are part of the tool-independent *bflow* Hive*. With the help of these plug-ins, modeling tool developers can include the functionality described in this article into their own Eclipse-based modeling tool. The easiest way for a user to profit from the features provided by the *bflow* Hive* in the modelling language of his or her choice is to download the most recent version of the *bflow* Toolbox* from the web site mentioned above and to use Eclipse's update mechanism for adding support for other modelling languages.

We are looking forward to reports from people who made use of the *bflow* Hive* into their tools, developed file exports, imports or add-ons using our mechanism or improved the source-code of the *bflow* Hive*.

Any questions related to our plug-ins are welcome to `bflow@bflow.org`.

References

1. Böhme, C., Hartmann, J., Kern, H., Kühne, S., Laue, R., Nüttgens, M., Rump, F.J., Storch, A.: bflow* Toolbox - an open-source business process modelling tool. In: Proceedings of the Business Process Management 2010 Demonstration Track. (2010) 46–51
2. Hannebauer, C., Book, M., Gruhn, V.: An exploratory study of contribution barriers experienced by newcomers to open source software projects. In: Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering. CSI-SE 2014, New York, NY, USA, ACM (2014) 11–14
3. Rieke, T., Winkelmann, A.: Modellierung und Management von Risiken. Ein prozessorientierter Risikomanagement-Ansatz zur Identifikation und Behandlung von Risiken in Geschäftsprozessen. Wirtschaftsinformatik **50** (2008) 346–356
4. Arendt, T., Taentzer, G.: A tool environment for quality assurance based on the Eclipse Modeling Framework. Automated Software Engineering **20** (2013) 141–184
5. Müller, C.: Generation of EPC based simulation modells. In: 25th European Conference on Operational Research. (2012)
6. Schneider, S., Laue, R., Storch, A., Mütze-Niewöhner, S.: Graphical modelling and simulation of product development projects. In: Proceedings of the 2012 European Simulation and Modelling Conference ESM'2012, EUROSIS (2011) 285–292
7. Gruhn, V., Laue, R.: A heuristic method for detecting problems in business process models. Business Process Management Journal **16** (2010) 806–821
8. Aysolmaz, B., Demirörs, O.: Automated functional size estimation using business process models with UPROM method. In: 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement. (2014) 114–124