Petriflow language and Netgrif Application Builder

Gabriel Juhás[1,2], Tomáš Kováčik[1,2], Jakub Kovář[1,2], Martin Kranec[1,2] and Ľuboš Petrovič[2]

[1]*Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovičova 3, 812 19 Bratislava, Slovakia*

[2]*NETGRIF, s.r.o., Slávičie údolie 106, 811 02 Bratislava, Slovakia*

Abstract

This paper summarizes the capabilities of the Petriflow language - a Petri net based modeling language for the creation of executable process-driven applications. Furthermore, the Netgrif Application Builder - a tool for creating Petriflow models is presented.

Keywords

Bussiness Process Design, Petri nets, Petriflow

1. Introduction

The Netgrif Application Builder (NAB) is the tool for building process-driven applications using the Petriflow language. NAB is a public web application available at https://builder.netgrif.com, where users can quickly design their own application process models or import models from any BPMN model. Petriflow is a high-level programming language for process-driven application development based on Petri nets.

2. Petriflow language

To model a control flow of a workflow process, place/transition Petri nets have been chosen. This choice was motivated by comparisons of Petri nets to other formalisms, such as BPMN, that are beyond the scope of this paper. Such comparisons have been done in the past, by many authors, such as [1] [2] [3] or [4].

To make the reflection of reality more accurate, Petriflow combines place/transition Petri Nets and some of their extensions:

- reset arcs [5], inhibitor arcs [6] and read arcs [5] were used to increase the expressiveness of place/transition Petri nets;

Proceedings of the Demonstration Resources Track, Best BPM Dissertation Award, and Doctoral Consortium at BPM 2021 co-located with the 19th International Conference on Business Process Management, BPM 2021, Rome, Italy, September 6-10, 2021

gabriel.juhas@stuba.sk (G. Juhás); tomas_kovacik@stuba.sk (T. Kováčik); jakub.kovar@stuba.sk (J. Kovář); martin.kranec@stuba.sk (M. Kranec); petrovic@netgrif.com (Petrovič)

0000-0001-8302-5112 (G. Juhás); 0000-0003-3157-8609 (T. Kováčik); 0000-0002-7775-3698 (J. Kovář); 0000-0002-9432-1344 (M. Kranec)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

- roles were added to define who can assign an enabled transition and who can execute a transition of the net;
- data variables were added to Petriflow to model attributes of the process instances, they can be associated with any task;
- actions were introduces to specify reactions to various events on the process model and data variables;

These extensions were introduced over time to enable the specification of executable enterprise applications, starting from place/transition Petri nets [7].

Importantly for transitions in Petri nets modelling a process, we use interval semantics, which defines the start and finish of a transition [8]. In this way, transitions can in a natural way model tasks. A state of a Petriflow process is given by the marking of places, a function indicating for each transition whether it is being executed and by which actor, and the values of the data variables.

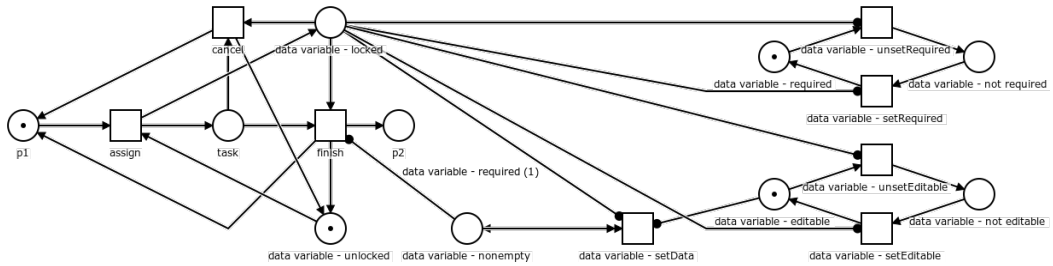

Figure 1: Lifecycle of a task represented by a Petri net

The complex behaviour of a task, including the locking of data variables and the effects of the required property on the ability to finish a task, can be modelled by an underlying Petri net as seen in figure 1. The required property is modelled by a variable arc weight determined by a place reference - the weight is equal to the marking of the referenced place. The finish event can therefore happen only if the referenced required data variable has some value set.

While many of the aspects of the Petriflow language (such as data, or roles) could be formalised in a rigorous way, we chose to forgo this purity in formalisation and chose instead to define the language as an XML notation, similarly to BPMN. A Petriflow process can be understood as a class (in the context of object-oriented programming) enriched by a workflow process that defines a life cycle of the class instances - cases [9].

3. Netgrif Application Builder

NAB is composed of several modules that help in different stages of application development:

- Process modeler - an editor for modeling processes based on Petri nets
- Role editor - an editor for creating and editing roles as well as specifying the permissions associated with them for individual tasks or the entire process instance

- Data editor - an editor for creating and editing data variables
- Form builder - an editor for creating and editing forms of specific tasks
- Actions editor - an editor for creating and editing actions (pieces of code in the Groovy scripting language)
- Simulation mode - simulation of the modeled Petri net either task by task or event by event
- BPMN model import - a utility that transforms BPMN files into executable Petriflow applications
- Petriflow model export - transforms the Petriflow model into its XML representation, which can be deployed into the Application Engine

3.1. Process Modeler

In the Process Modeler you can model business processes by defining tasks and their routing and simulate the modeled processes by executing sequences of tasks. As a modeling formalism for processes, Petriflow language uses Petri nets that consist of state variables represented by places, tasks represented by transitions and their interconnections, that define rules for the executability of the various tasks based on the state variables [4].

3.2. Role Editor

The role editor can be used to enrich the Petriflow processes with permission granting or permission revoking roles. Roles can be understood as sets of application users, shared between all instances of the same process model. The role can then be associated with any of the Petriflow events with a positive or a negative relation. When a role is associated with an event, such as the assign event of some task, in a positive way, then only members of the role are allowed to perform this event, in our example, this means that only these users would be able to assign the specific task. If the roles are associated in a negative way, only the users that are not members of the specified role will be able to trigger the associated event.

An instance scoped variant of roles exists - the users list data variable. This data variable also contains a set of users but its value can differ for each process instance. The role editor can be used to assign permissions to roles, as well as users list data variables.

3.3. Data Editor

The data editor manages data variables used in the processes. The Petriflow language supports all the basic types of data variables, including text, numbers, date, date-time, enumerations and choices, files, images and many others. The data variables are an important part of the Petriflow language as they hold valuable business-specific data, or are used to control the direction of the workflow with features such as variable arc weights or inter-process communication.

3.4. Form Builder

The Form Builder is used to associate existing or new data variables with the individual tasks of the model. The association creates a form that the user then interacts with when they execute the

appropriate task. Associating the data variables with the task is done by drag-and-dropping the variables into a grid to specify their layout and it can then be further customized by specifying different attributes, such as placement in the form grid layout, choosing the appropriate view, such as different types of check-boxes, and determining, whether the data fields are editable, visible, required or hidden.

A special type of data variable - the task reference, can be used to transparently embed forms of one transition into another, allowing us to create highly structured, nested and reusable forms which enhance the functionality of the application.

3.5. Actions Editor

In the Action Editor we can define reactions to events of the process instances, their tasks, and data variables. Actions are based on the Groovy programming language extended with various methods that grant the programmers access to the Petriflow actions API. [10]

The various types of events that we can react to include creation of the process instance, assignment of a task to a user, cancellation of a task, completion (finish) of a task, and the change of a data field value. Actions can be used to create new process instances, assign tasks to users, recalculate data variable values, or to hide/show data fields in task forms. These effects are themselves events and can therefore trigger other actions in turn. [10]

Furthermore, actions have access to the entire Application Engine, so they can also use integrations to third-party web services, send emails or dynamically create files for the users of the application to download.

3.6. Simulation Mode

In order to test the model and to visualize the workflow, the Netgrif Application Builder contains a simulation mode. It allows us to see the Petri net model, to step through it, by executing the individual transitions or the their events and to see the changes this causes to the marking of the net.

The simulation does not encompass all the aspects of the Petriflow language and is limited only to simulating the Petri net model. A complete simulation including roles, actions and data variables can be performed on a deployed model in a test environment of an Application Engine.

3.7. BPMN Model Import

The Netgrif Application Builder includes the functionality of importing BPMN models using a modification of an existing algorithm, taking into consideration the interval semantics of tasks, to convert the BPMN language to the formalism of Petri nets.

3.8. Petriflow Model Export

The Petriflow language has an XML based syntax and the Application Builder aims to be fully functional visual based IDE for the Petriflow language. Our goal is to achieve tool cross-compatibility where any builder can open any Petriflow model, that can be interpreted and executed by any engine, regardless of the models origin.

4. Materials

This video demonstration of all the core aspects of the Petriflow platform can be used as a good starting point for designing and deploying your first process-driven application https://www.youtube.com/watch?v=iU1QGPUnXUs.

Alternatively we have a website dedicated to on-boarding materials and examples, that you can find inspiration and guidance on https://netgrif.com/started/.

References

[1] W. van der Aalst, On the representational bias in process mining, in: 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, 2011. URL: https://doi.org/10.1109/wetice.2011.64. doi:10.1109/wetice.2011.64.

[2] A. Koschmider, A. Oberweis, W. Stucky, A petri net-based view on the business process life-cycle, Enterprise Modelling and Information Systems Architectures (2018) Vol 13 (2018). URL: https://www.emisa-journal.org/emisa/article/view/192. doi:10.18417/EMISA.SI.HCM.4.

[3] W. M. P. van der Aalst, Business process management as the "killer app" for petri nets, Software & Systems Modeling 14 (2014) 685–691. URL: https://doi.org/10.1007/s10270-014-0424-2. doi:10.1007/s10270-014-0424-2.

[4] J. Desel, G. Juhás, "what is a petri net?" informal answers for the informed reader, in: Unifying Petri Nets, Springer Berlin Heidelberg, 2001, pp. 1–25. URL: https://doi.org/10.1007/3-540-45541-8_1. doi:10.1007/3-540-45541-8_1.

[5] R. Lorenz, J. Desel, G. Juhás, Models from scenarios, in: Transactions on Petri Nets and Other Models of Concurrency VII, Springer Berlin Heidelberg, 2013, pp. 314–371. URL: https://doi.org/10.1007/978-3-642-38143-0_9. doi:10.1007/978-3-642-38143-0_9.

[6] G. Juhás, R. Lorenz, S. Mauser, Complete process semantics for inhibitor nets, in: Petri Nets and Other Models of Concurrency – ICATPN 2007, Springer Berlin Heidelberg, 2007, pp. 184–203. URL: https://doi.org/10.1007/978-3-540-73094-1_13. doi:10.1007/978-3-540-73094-1_13.

[7] M. Mladoniczky, G. Juhás, J. Mazári, T. Gazo, M. Makán, Petriflow: Rapid language for modelling petri nets with roles and data fields, Algorithms and Tools for Petri Nets 45 (2017).

[8] M. Alqarni, R. Janicki, On interval process semantics of petri nets with inhibitor arcs, in: Application and Theory of Petri Nets and Concurrency, Springer International Publishing, 2015, pp. 77–97. URL: https://doi.org/10.1007/978-3-319-19488-2_4. doi:10.1007/978-3-319-19488-2_4.

[9] G. Juhás, Process-driven programming, 2021. URL: https://netgrif.com/process-driven-programming/.

[10] J. Mazári, G. Juhás, M. Mladoniczky, Petriflow in actions: Events call actions call events, Algorithms and Tools for Petri Nets (2018) 21–26.