Годишник на секция "Информатика" **Annual of "Informatics" Section**
Съюз на учените в България **Union of Scientists in Bulgaria**
Том 5, 2012, 111-119 **Volume 5, 2012, 111-119**

VERIFYING *BPMN* PROCESSES USING GENERALIZED NETS

Pavel Tcheshmedjiev

Bioinformatics and Mathematical Modelling Department
Institute of Biophysics and Biomedical Engineering
Bulgarian Academy of Sciences
email: pavel@biomed.bas.bg

Abstract: The present paper presents a new approach for verifying Business Process Modelling Notation (BPMN) processes using Generalized Nets (GN). This mapping enables the GN representation of business process diagrams and execution code with test case verification thus combining the convenience of business friendly visual diagramming with the execution, simulation and verification capabilities of the generalized nets.
Keywords: Verification, Business process management, Business process modelling notation, Generalized nets.

1. Introduction

Verification plays a very important role in the software engineering process and is a key repeating phase in the continuous integration of the software product. The main goal of the verification process is to ensure that the implemented use cases meet the defined requirements in terms of provided functionality and correct behavior, result and error handling. Thus verification together with validation is a critical factor for providing and maintaining software quality.

Generally, two approaches are used for software verification – black-box and grey-box testing. When black-box testing is used, the software is considered a 'black box', and its functionality is tested without any knowledge of the internal implementation.

Grey-box testing requires knowledge of the internal structures, data and algorithms for purposes of designing tests. The tester is not required to have full access to the software's source code. Typical example is when executing integration testing between two modules of code written by different parties, where only the interfaces are exposed for test.

The primary goal of Business Process Modelling Notation (BPMN) is to provide a notation that is readily comprehensible to all business users, to the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation [1].

BPMN specification defines the notation and semantics of a Business Process Diagram (BPD) and represents the best practices within the business modelling community. The intent behind BPMN is to standardize the multiple different notations and viewpoints in a uniform business process modelling notation. In this way, BPMN provides a simple means of communicating process information to other business users, process implementers, customers and suppliers.

The gap between business and technical people needs to be bridged with a formal mechanism that maps the appropriate visualization of the business processes (a notation) to the appropriate execution format (a BPM execution language) for these business processes. The BPMN specification proposes BPEL4WS as the primary languages that BPMN will map to.

In the present paper, we use the existing BPMN mapping to generalized nets [2] and use GNs as a formal mathematical model with well defined execution rules and well developed mathematical foundation to introduce process verification. We will introduce a verification approach and give an example based on OMG sample BPMN support process [4] with business process engine support and web service calls [5]. Introducing GN models for verifying software products can reduce the complexity of writing automation scripts and also help business analysts to contribute to both defining the right use cases and the respective verification test cases. GN models can be used for both black-box and grey-box testing because there are no restrictions with monitoring GN model execution.

2. The concept of generalized nets

Generalized nets are extensions of the Petri nets and their other modifications. They are a tool intended for detailed modelling of parallel processes.

A generalized net is a collection of *transitions* and *places* ordered according to some rules (see Figure 1).

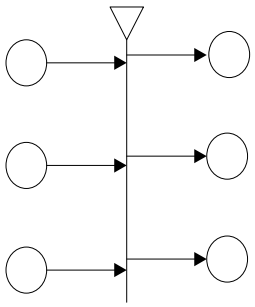

Figure 1. A GN transition

The places are marked by circles. The set of places to the left of the vertical line (the transition), are called input places and the ones that are to the right are called output places. For each transition there is an *index matrix* with elements – *predicates*. Some GN-places contain *tokens* which are the dynamic elements and carriers of information, entering the net

with initial characteristics and obtaining new ones during their movement within the net. Tokens proceed from the input to the output places of the transitions if the predicate corresponding to these places in the index matrix is evaluated as “true”. Every token has its own identifier and collects its own history that could influence the development of the whole process modelled by the GNs.Two time-moments are specified for the GNs: startup and termination of functioning, respectively.

A GN can have only a part of its components. In this case it is called a reduced GN. Here we shall give the formal definition of a reduced GN without temporal components, place and arc capacities, and token, place and transition priorities.

Formally, every transition in the used below reduced GN is described by the triple:

$$Z = \langle L', L'', r \rangle,$$

where:

(a) L' and L'' are finite, non-empty sets of places (the transition’s input and output places, respectively). For the above described transition these are $L' = \{l'_1, l'_2, \ldots, l'_m\}$ and $L'' = \{l''_1, l''_2, \ldots, l''_n\}$.

(b) r is the transition’s *condition* determining which tokens will pass (transfer) from any of the inputs to any of the outputs. it has the form of an Index Matrix (IM):

$$r = \begin{array}{c|c} & l''_1 \quad \ldots \quad l''_j \quad \ldots \quad l''_n \\ \hline l'_1 & \\ \ldots & r_{i,j} \\ l'_i & (r_{i,j} \; - \; predicate) \\ \ldots & (1 \le i \le m, \; 1 \le j \le n) \\ l'_m & \end{array}$$

Here, $r_{i,j}$ is the predicate that corresponds to the i-th input and j-th output place. When its truth value is ‘true’, a token from the i-th input place is eligible to transfer to the j-th output place; otherwise, this is not possible;

The ordered four-tuple

$$E = \langle A, K, X, \Phi \rangle$$

is called a *reduced Generalized Net* if:

(a) A is the set of transitions;

(b) K is the set of the GN’s tokens;

(c) X is the set of all initial characteristics which the tokens can obtain on entering the net;

(d) Φ is the characteristic function that assigns new characteristics to every token when they make the transfer from an input to an output place of a given transition.

A lot of operations (e.g., union, intersection, etc.), relations (e.g., inclusion, coincidence, etc.) and operators are defined over the GNs. Operators change the GN-forms, the strategies of token transfer and other. There are six types: global, local, hierarchical, reducing, extending and dynamic operators.

3. Verification approach

Every business process that will be modelled with Business Process Modelling Notation and executed on BPMN engine has to be verified in order to pass all phases of acceptance testing and be finally deployed to production environment. Typically during the business analyses phase when the business process itself is being created according to the specific use case there should be defined the respective test cases for acceptance. For verification purposes it should be clear the possible input data for the process and what should be the expected output [6]. These test cases can be defined for the whole process if using black-box testing approach or for specific process phases as in grey-box testing approach.

When a given BPMN process is represented as GN model using the proposed mapping [3] we can easily define initial properties for the tokens in the input place - $L_{i'}$ that will represent test case input data, start the simulation of the GN and monitor the properties of the tokens in the output position $L_{i'''}$. As shown on Fig. 3 tokens in intermediate positions $L_{i''}$ can also be monitored for preparing full verification profile.

4. Sample BPMN with GN verification

For illustrating the benefits of the proposed approach we will use OMG standard BPMN example process [5] for realizing automated incident management process (see Fig. 2).

The business process represents a model of the actions related to assignment and processing of support tickets – we would like to introduce the assignment of tickets to 1st and 2nd level support agents by a trouble ticket system, which takes the role of the process engine. That support management system can receive and parse emails sent by the account manager and opens a ticket for it. If the 1st level support agent decides that this is a 2nd level issue, he does so by documenting his decision and completing the assigned task "edit 1st level ticket". The trouble ticket system then routes the ticket to the 2nd level support agent. When that agent has finished, the issue can be declared to be fixed in the next software release. This can be realized with service call to external backend system. In the end, the trouble ticket system will send an email to the account manager, containing the results of the incident management, and close the ticket. The account manager can then explain the solution to the customer based on the information in the ticket.

The described business process can be executed on process engine where the following language mappings are used the following XML tags:

- Process is defined using process tag with executable property:

```
<process isExecutable="true" id="WFP-1">
```

- Issue data is defined using *dataInput* tag

```
<dataInput itemSubjectRef="tns:IssueItem"
id="IssueDataInputOfProcess" />
```

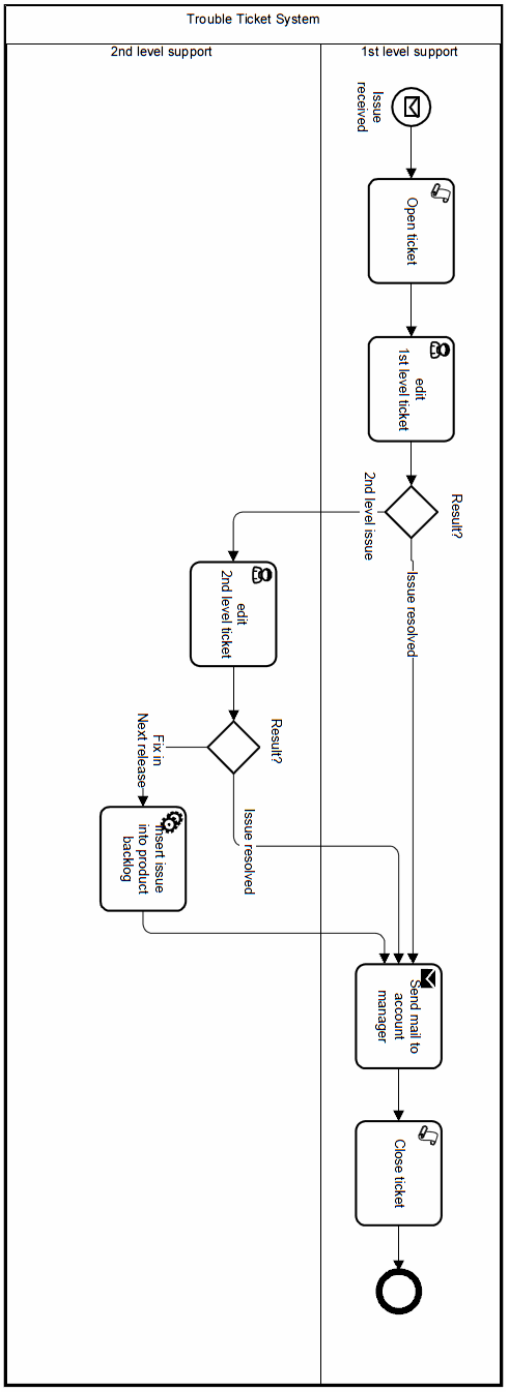

Figure 2. OMG ticket support process

- Process lanes are defined using the *laneSet*, *lane* and *flowNodeRef* tags:

```
<laneSet id="ls_1-1">
<lane name="1st level support"
partitionElementRef="tns:FirstLevelSupportResource" id="_1-9">
<flowNodeRef>_1-13</flowNodeRef>
<flowNodeRef>_1-26</flowNodeRef>
<flowNodeRef>_1-77</flowNodeRef>
</lane>
<lane name="2nd level support"
partitionElementRef="tns:SecondLevelSupportResource" id="_1-11">
<flowNodeRef>_1-252</flowNodeRef>
<flowNodeRef>_1-303</flowNodeRef>
<flowNodeRef>_1-325</flowNodeRef>
</lane>
</laneSet>
```

- Input/Output data is defined using *ioSpecification*, *dataInput* and *dataOutput* tags:

```
<ioSpecification>
<dataInput itemSubjectRef="tns:TicketItem" id="TicketDataInputOf_1-
77" />
<dataOutput itemSubjectRef="tns:TicketItem"
id="TicketDataOutputOf_1-77" />
<inputSet>
<dataInputRefs>TicketDataInputOf_1-77</dataInputRefs>
</inputSet>
<outputSet>
<dataOutputRefs>TicketDataOutputOf_1-77</dataOutputRefs>
        </outputSet>
```

- I/O data associations are defined using *dataInputAssociation*, *dataOutputAssociation*, *sourceRef* and *targetRef* tags

```
<dataOutputAssociation>
<sourceRef>TicketDataOutputOf_1-77</sourceRef>
<targetRef>TicketDataObject</targetRef>
</dataOutputAssociation>
```

- Sequence and conditions are defined using the sequence flow and *conditionExpression* tags.

```
<sequenceFlow sourceRef="_1-303" targetRef="_1-150"
name="Issue resolved" id="_1-406">
<conditionExpression xsi:type="tFormalExpression">
${getDataObject("TicketDataObject").status == "Resolved"}
</conditionExpression>
</sequenceFlow>
```

On Fig. 3 is shown the GN model of the described business process.

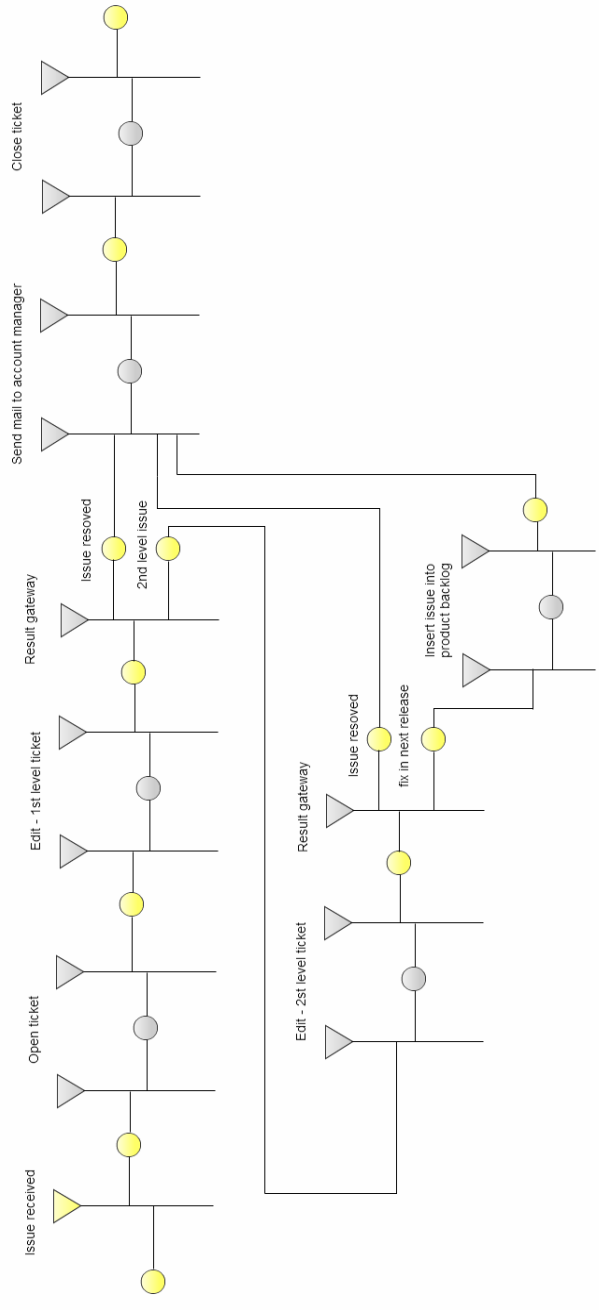

Figure 3. GN model of the ticket support process

Every BPMN sequence flow is modeled as transition and all input/output data as tokens. In our case the data token represents the ticket for the issue with attributes – id, description, status, date, history. In order to validate and verify the process it is sufficient to define input transitions with respective token values and the expected result and output transition. Thus the whole use case can be verified by using the predefined test cases – input data and expected values or error conditions.

These are example test cases that will be used for verifying the process and related use case:

Test case	Description	Input value	Expected result	Related GN transition
1	Validate first level support sequence	Sample ticket	- Ticket is not sent to second level support, - Next email sent to account manager, - Ticket closed.	Issue received, Open ticket, Edit-1st level support, Result gateway, Send mail, Close ticket.
2	Validate second level support sequence	Sampe ticket	- Ticket is sent to second level support, - Next email sent to account manager, - Ticket closed.	Issue received, Open ticket, Edit-1st level support, both result gateways, Edit 2nd level support, Send mail, Close ticket.
3	Validate product backlog	Sample ticket	- Ticket is sent to second level support and backlog item created, - Next email sent to account manager, - Ticket closed.	Issue received, Open ticket, Edit-1st level support, both result gateways, Edit 2nd level support, Insert issue into backlog, Send mail, Close ticket.

When the GN model is started on the simulator we need to define the input tokens and track and trace their way to the final place. When the token passes through transitions its attribute "status" is changed and logged. Depending on the test case – with or without second level support, or with or without inserting issue into the backlog, the sample token should or should not pass through the respective GN transitions. All scenarios should end with token passing through the Close ticket transition. Test case is considered to pass verification if the input token passes through all predefined transitions and status is changed accordingly till it becomes closed. All test cases can be executed on GN simulator as part of the software continuous integration procedure.

5. Conclusion

Mapping BPMN to GN and verifying BPMN processes using GNs provides powerful combination of business friendly visual diagramming with powerful execution/simulation capabilities of a formal mathematical model. Generalized Nets provide full representation the complete BPD element set, they are simple and flexible means to model use cases and

test cases. GN and BPMN are similar in scope but have different focus. BPMN and BPEL focus on the visual and business representation while GN focus on execution and simulation which is the main advantage to be used when process verification should be performed.

Representing and verifying business processes as GNs can be automated using the available GN tools and execution environment. We can simulate and validate BPMN in the currently available GN environment, which can be used for black-box and grey-box acceptance testing. It can also be used for analyses of the behaviour of complicated business processes as is the case of concurrent process instances;

Another advantage of the approach is that it helps business and technical people to cooperate using the same tools – BPMN and GN models. This will allow collaborating parties to have a better understanding of the process, the test conditions and the verification.

Acknowledgments

This work has been supported by the Bulgarian National Science Fund, Grant DMU-03-38 "Management and modelling of biochemical, medical processes and information with the application of generalized nets and linked data".

References

[1] Business Process Modelling Specification, OMG, January 2009, http://www.omg.org/spec/BPMN/

[2] Atanassov, K., Generalized Nets, World Scientific, Singapore, 1991.

[3] Ivanov, I., B. Kolev, Mapping of Business Process Modelling Notation Elements to Generalized Nets: Complete Element Set, Tenth Int. Workshop on Generalized Nets Sofia, 5 December 2009, 31–40.

[4] BPMN example, OMG, June 2010 http://www.omg.org/spec/BPMN/2.0/examples/PDF

[5] Todorova. M. Formal Specification of WEB Services by Means of Generalized Nets with Stop-Conditions, 5th International Conference Information Systems & Grid Technologies, 28–29 May 2011, 215–227.

[6] Todorova, M. Implementation of an Approach for Verification of Procedural Programs, Global Science and Technology Forum: International Journal on Computing, Vol. 2, No. 2, June 2012, 70–75, DOI: 10.5176/2010-2283_2.2.170.