

Mapping of Business Process Modelling Notation Elements to Generalized Nets: Complete Element Set

Ivaylo Ivanov¹ and Boyan Kolev²

¹ – SoftConsultGroup Ltd

Dedeagach Str, Bl. 60A, c-x Strelbishte, Sofia-1408, Bulgaria

e-mail: *ivaylo.ivanov@softconsultgroup.com*

² – Centre of Biomedical Engineering, Bulgarian Academy of Sciences

Acad. G. Bonchev Str., Bl. 105, Sofia-1113, Bulgaria

e-mail: *bik@clbme.bas.bg*

Abstract: This paper presents a mapping of business process modelling notation (BPMN) elements in terms of generalized nets (GN). This mapping enables the GN representation of business process diagrams thus combining the convenience of business friendly visual diagramming with the execution/simulation capabilities of the generalized nets.

Keywords: Business process management, Business process modelling notation, Generalized nets

1 Introduction

Generalized Nets (GNs, see [2]) are defined as extensions of the ordinary Petri nets and other modifications of theirs, but in a way that is principally different from the ways of defining the rest types of Petri nets. The additional components in the GN definition give more and larger modelling possibilities and determine the place of GNs among the separate types of Petri nets, similar to the place of the Turing machine among the finite automata.

The GNs are a suitable tool for describing parallel processes flowing in real-time, because they give the possibility to represent the process from analytical point of view (using GN-token characteristics), as well as from logical point of view (using the GN-transition condition predicates).

The Business Process Management Initiative (BPMI) has developed the Business Process Modelling Notation (BPMN) standard. The primary goal of 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].

This BPMN specification defines the notation and semantics of a Business Process Diagram (BPD) and represents the amalgamation of 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, [1]

Business people are very comfortable with visualizing business processes in a flow-chart format. There are thousands of business analysts studying the ways companies work and defining business processes with simple flow charts. This creates a technical gap between the format of the initial design of business processes and the format of the languages or environments that will execute these business processes. This gap 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.

Meanwhile, the structure and vocabulary of BPMN differs from BPEL4WS. BPMN allows flexible and free-form methods of connecting activities through Sequence Flow. Furthermore, BPMN is cyclical in that it allows Sequence Flow to connect to upstream objects so that a modeller can easily visualize looping situations. BPEL4WS has a much more structured form of creating a process flow. The flow activity in BPEL4WS does allow some flexibility with its link elements, but is acyclical. Thus, there is not going to be a one-to-one mapping of the BPMN elements to the BPEL4WS elements, without restricting the connection capability of BPMN, [1].

The fact is that BPMN sets high expectations about the environment that executes BPDs and BPEL4WS fails to meet some of them. Other more flexible and more powerful execution environments are required.

We propose BPMN mapping to generalized nets (GN), because GN is a formal mathematical model with well defined execution rules and well developed mathematical foundation. Thus, we can utilize the power of formal mathematical models of GNs with the visualization / modelling power of BPMN.

The core BPD elements support the requirement of a simple notation. These are the elements that define the basic look-and-feel of BPMN. Most business processes can be modelled adequately with these elements. The complete set of BPD elements consists mostly of different kinds of variations of the basic ones including both graphical and non-graphical attributes, providing more detailed information about the business process. Their GN equivalents will not differ significantly in graphical structure from the basic ones; they will rather complete the logical components of the GN like predicate conditions, place capacities, transition types, etc. In this paper we will present a mapping of the complete BPD element set to GNs. The rest of the paper is organized as follows: the next chapter defines the scope of BPMN to GN mapping; the third chapter presents the basic BPD elements and their GN equivalents; the fourth chapter describes the detailed elements along with the corresponding GN components.

2 Scope of BPMN to GN mapping

According to [1], BPMN is constrained to support only the concepts of modelling that are applicable to business processes. This means that other types of modelling done by organizations for business purposes will be out of scope for BPMN. For example, the modelling of the following will not be a part of BPMN:

- Organizational structures and resources,
- Functional breakdowns,
- Data and information models,
- Strategy,
- Business rules.

Those areas will be out of scope for GN representations of BPMN also. There are three basic types of sub-models within an end-to-end BPMN model:

- Private (internal) business processes
- Abstract (public) processes
- Collaboration (global) processes, [1].

GN representation of BPMN will only cover a single executable private business process. If a BPMN diagram depicts more than one internal business process, then there will be a separate mapping for each of the internal business processes.

Private (Internal) Business Processes are those internal to a specific organization and are the types of processes that have been generally called workflow or BPM processes, [1]. A single private business process may be mapped to one or more GNs. Mapping Abstract and Collaboration part of the BPMN diagram is out of scope for this paper.

3 GN Representations of Core BPD Element Set

One of the drivers for the development of BPMN is to create a simple mechanism for creating business process models, while at the same time being able to handle the complexity inherent to business processes. The approach taken to handle these two conflicting requirements was to organize the graphical aspects of the notation into specific categories. This provides a small set of notation categories so that the reader of a BPMN diagram can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look and feel of the diagram, [1]. The proposed in this paper mapping allows a BPD diagram to be transformed into a GN. GN tokens correspond to process instances. Each GN token enters the GN with a mandatory characteristic `process_instance_id`, which identifies the process instance the GN token corresponds to. Multiple tokens with the same `process_instance_id` represent process branches, running in parallel. Here are the most common basic BPD elements and their representations in terms of generalized nets:

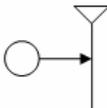
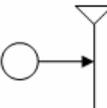
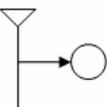
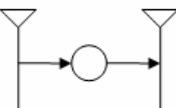
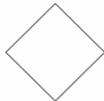
Element	BPMN	GN
Start Event		
Intermediate Event		
End Event		
Swimlanes		N/A
Artifacts		N/A
Element	BPMN	GN
Activity		
Gateway		
Sequence flow		
Message flow		N/A
Association		N/A

Table 1: Mapping of BPD elements to GN constructs

Some of the BPD elements have no GN representation as their semantics is out of scope of the BPMN to GN mapping as discussed in the second section. In this paper we will discuss only those BPD elements that have GN representation.

3.1. Flow Objects

3.1.1. Events

An *event* is something that “happens” during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). There are three types of events, based on when they affect the flow: Start Event, Intermediate Event and End Event.

- A *Start Event* indicates where a particular process will start. The corresponding GN structure is an input place, followed by a transition. The GN token, representing the modelled process, enters the generalized net through this place. Whenever a token enters the input place, a new `process_instance_id` is created. The token is marked with the new `process_instance_id`.
- An *Intermediate Event* occurs between a *Start Event* and an *End Event*. It will affect the flow of the process, but will not start or (directly) terminate the process. The corresponding GN structure is an input place, followed by a transition. A GN token enters the generalized net through this place with the `process_instance_id` of a process, which has already started, i.e. at least one token with the same `process_instance_id` exists in the GN.
- An *End Event* indicates where a particular process will end. The corresponding GN structure is a transition, followed by an output place. The GN token, representing the modelled process, leaves the generalized net through this place. Whenever a token enters the output place, all the tokens within the GN that have the same `process_instance_id` are deactivated.

3.1.2. Activities

An activity is a generic term for work that a company performs. An activity can be atomic or non-atomic (compound). The corresponding GN structure consists of two transitions with a place between them. The transitions represent the conditions for starting and ending the activity, while the place represents the activity itself, i.e. a GN token stays in the place while the activity is being performed.

- *Task* (atomic activity) - used when the work in the Process is not broken down to a finer level of Process Model detail.
- *Sub-process* (non-atomic) – this is a compound activity that is included within a Process. It is compound in that it can be broken down into a finer level of detail (a Process) through a set of sub-activities. In GN terms this is equal to applying a hierarchical operator H_1 over the place, corresponding to the activity.

3.1.3. Gateways

A *Gateway* is used to control the divergence and convergence of multiple Sequence Flows. Thus, it will determine branching, forking, merging, and joining of paths. The corresponding GN structure is a single transition. Generally a gateway can have multiple incoming and outgoing flows, respectively the GN transition can have multiple input and output places.

3.2. Connection Objects

3.2.1. Sequence Flow

Sequence Flows are objects, used to connect events, activities and gateways to each other. They show the order that activities will be performed in a process. A process may block at a Sequence Flow in the case when the process has to wait for a certain condition (e.g. when joining two flows at a gateway). That is why the corresponding GN structure consists of a place with an input and an output arcs.

4 GN Representations of Complete BPD Elements

Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look and feel of the diagram, [1].

This chapter explains the extensions of the BPMN to GN mapping model that cover the complete BPD element set.

Only the BPD elements from the complete set that require additional modelling are described. All the others that are not mentioned have default GN representations described in the previous chapter.

4.1. Event variations

Events may differ by the types of trigger that define the cause of the event. Figure 2 depicts different types of event triggers in three columns showing whether the trigger is applicable for start, intermediate and end events respectively. In the corresponding GN structure the trigger type affects the predicate condition. In case of start and intermediate events the predicate generally either makes the token stay in its current place or lets the token move towards the upcoming sequence flow. In the case of end event, the event type defines the “Result” that is a consequence of a sequence flow ending; therefore the event type reflects on the characteristic function of the output place in the corresponding GN construct.

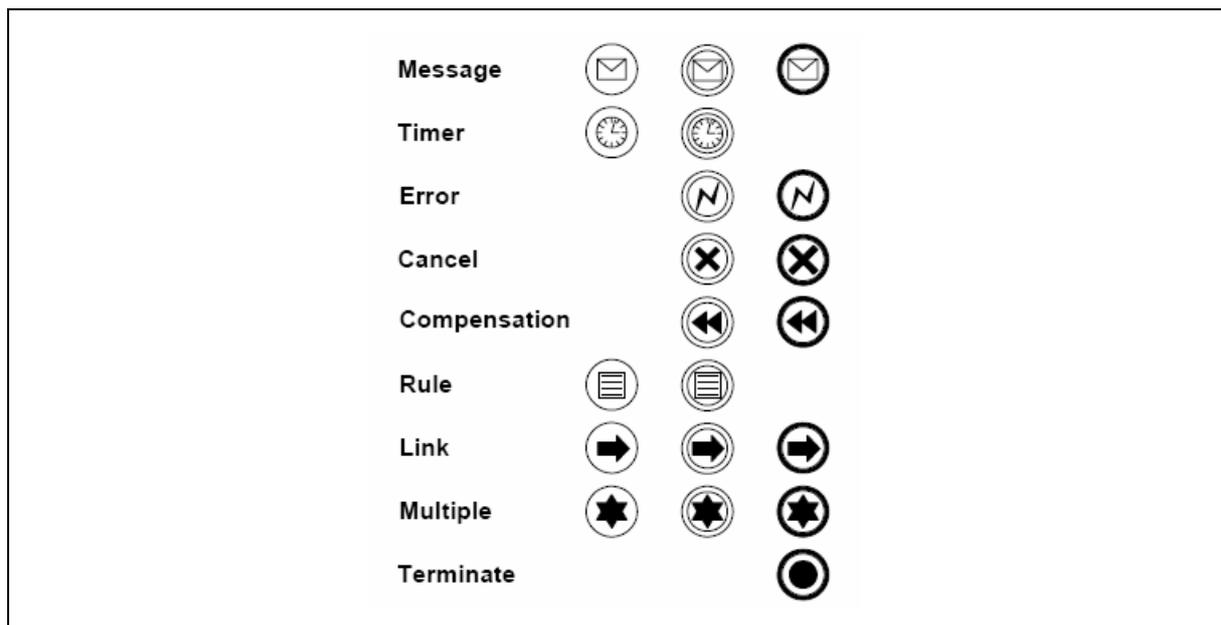


Figure 2: BPMN Event types

4.2. Collapsed and Expanded Sub-Process

These are ways to depict sub-processes without or with detailed information. As stated in 2.2, in GN terms sub-process is equal to applying a hierarchical operator H_1 over the place, corresponding to the activity.

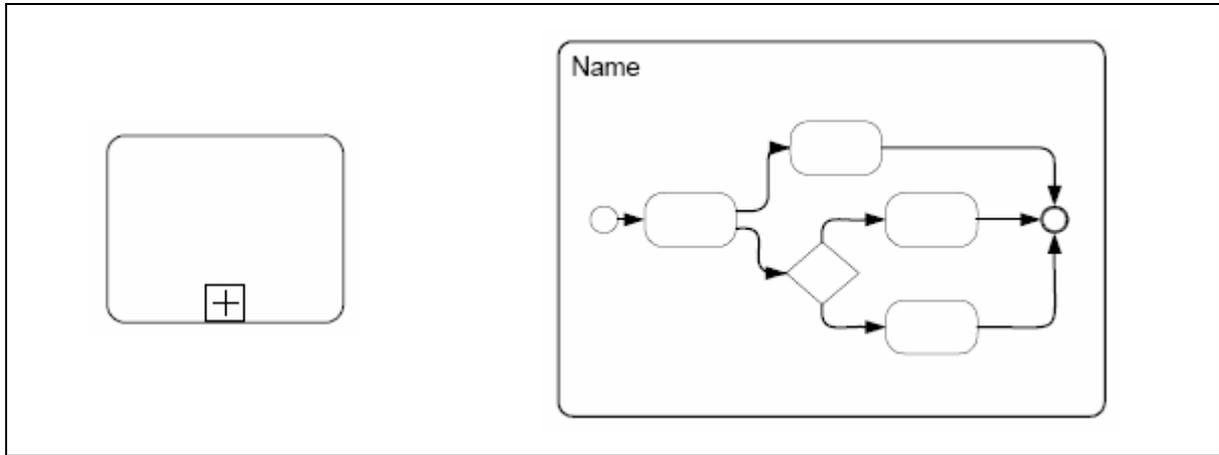


Figure 3: BPMN Collapsed and Expanded Sub-Process

4.3. Sequence Flow variations

When outgoing from an activity, sequence flows may be classified based on a condition for passing through the flow.

- **Conditional Flow** is a sequence flow that has a condition expression, evaluated at runtime, to determine whether or not the flow will be used (Figure 4.a).
- **Default Flow** is a sequence flow, used only if all the other outgoing conditional flows are not true at runtime (Figure 4.b).
- **Exception Flow** occurs outside the normal flow of the process and is based upon an exception intermediate event that occurs during the performance of the process (Figure 4.c).

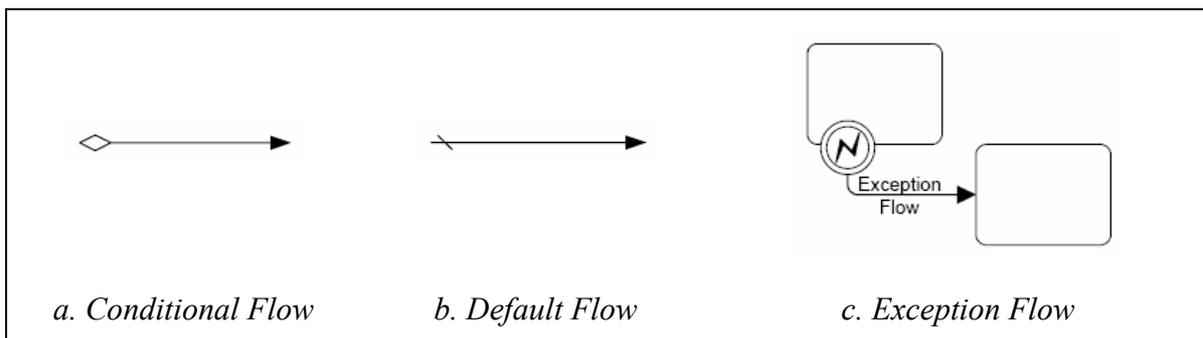


Figure 4: BPMN Sequence Flow Variations

In the corresponding GN structure flow conditions reflect on the transition predicates which let tokens move to the corresponding output place of the activity transition. For example, the following predicate matrix is for the second transition of an activity GN equivalent with four outgoing flows: two conditional (L_{C1} and L_{C2}), a default (L_D) and an exception (L_E) flows:

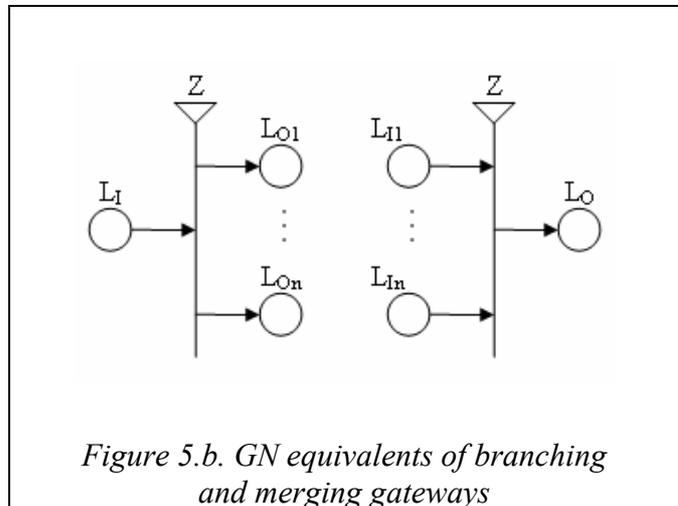
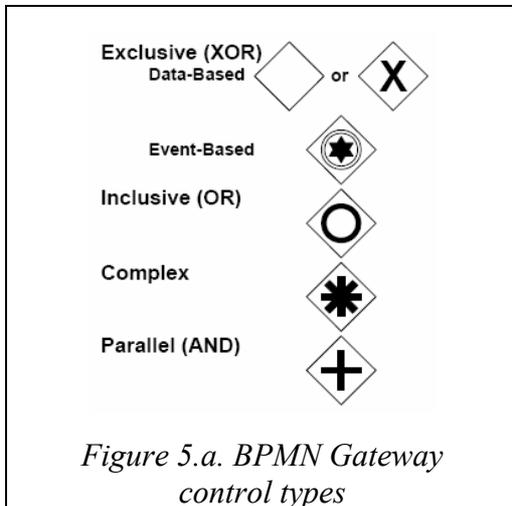
$$r = \frac{\quad}{L_{activity}} \left| \begin{array}{cccc} L_{C1} & L_{C2} & L_D & L_E \\ \hline W_{C1} \& \neg W_E & W_{C2} \& \neg W_E & \neg(W_{C1} \vee W_{C2} \vee W_E) & W_E \end{array} \right.$$

where:

- W_{C1} is the predicate, defined with the condition of the first conditional flow
- W_{C2} is the predicate, defined with the condition of the second conditional flow
- W_E = “an exception occurred during activity execution”
- $L_{activity}$ is the place, corresponding to the activity (see 2.2)

4.4. Gateway control types

Gateways perform flow control by determining the way of branching and merging sequence flows. Generally, there are four types of flow control (see Figure 5.a): XOR (exclusive decision and merging), OR (inclusive decision and merging), Complex (complex conditions and situations, e.g. 3 out of 5), and AND (forking and joining). Each type of control affects both the incoming and outgoing flow.



4.4.1. XOR Gateways

An XOR gateway represents exclusive decision. It restricts the flow in such way that exactly one of a set of alternatives may be chosen during runtime. There are two types of exclusive gateways:

- **Data-based:** The decision represents a branching point, where alternatives are based on conditional expressions contained within the outgoing sequence flow.
- **Event-based:** The alternatives are based on an event that occurs in that point of the process. The specific event, usually the receipt of a message, determines which of the paths will be taken.

For both data-based and event-based gateways, the predicate matrix of the corresponding GN transition consists of mutually exclusive predicate conditions.

4.4.2. OR Gateways

A branching OR gateway represents inclusive decision point, where alternatives are based on conditional expressions contained within the outgoing sequence flow. In some sense it is a grouping of related independent binary (yes/no) decisions. Since each path is independent, all combinations of the paths may be taken, from zero to all. A default condition could be used to ensure that at least one path is taken. The predicate matrix of the corresponding GN transition consists of independent predicate conditions.

A merging OR gateway is used to show the combining of two or more paths into one path. The corresponding GN transition lets the tokens pass unconditionally.

4.4.3. AND Gateways

A splitting AND gateway represents a forking of a process path into two or more parallel paths, where activities are performed concurrently rather than sequentially. A token in the corresponding GN construct splits to several tokens with the same `process_instance_id`, which continue their movement to all of the output places. The predicate matrix of the transition consists only of predicates with strict *true* value.

A joining AND gateway represents the combining of two or more parallel paths into one path (also known as AND-join or synchronization). An activation condition for the corresponding GN transition should be the existence in all input places of tokens with the same `process_instance_id`. All these tokens move to the output place, where they are merged into a single token.

4.5. Activity Looping

The attributes of task or sub-process determine whether it is repeated or performed once. An Activity Looping BPD element represents an activity, which can internally be repeated more than once. Tokens, which need to enter the activity, will have to wait for the token that is currently in the activity to finish with all the repetitions of the task. Fig. 6.b depicts the corresponding GN construct, extended with a place (L_{AL}), through which the active token loops and returns to the activity place (L_A). Places L_{ISF} and L_{OSF} represent respectively an incoming and an outgoing sequence flow. If there is a token, currently performing an activity loop, all pending tokens in L_{ISF} have to wait until the loop finishes and then one of them will be let in the activity. This is achieved by setting a higher priority to L_{AL} than to L_{ISF} and setting the capacity of place L_A to 1.

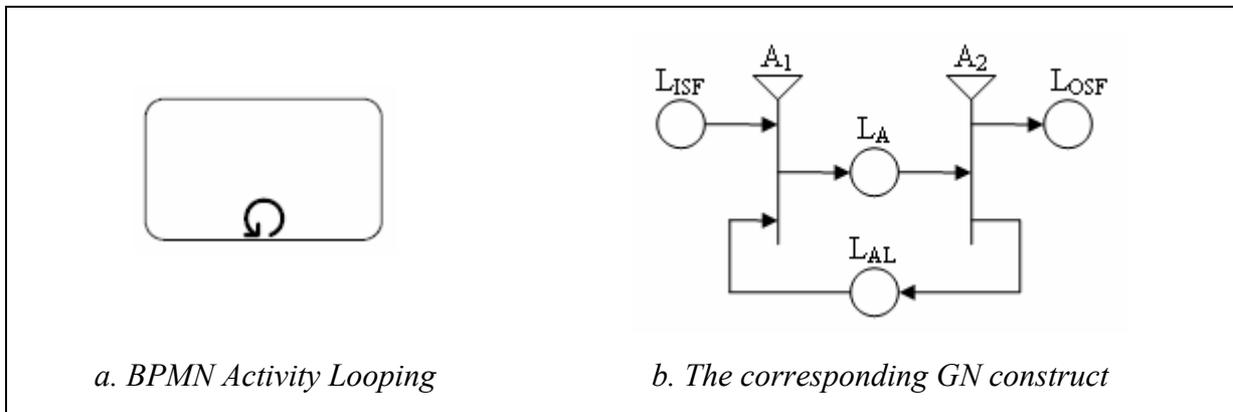


Figure 6: BPMN Activity Looping and its Corresponding GN Construct

4.6. Multiple Instances

A multiple instances activity can be enabled multiple times for a single process instance. A token, entering the activity object, is split into several tokens with the same `process_instance_id`, which perform the activity either concurrently or sequentially, depending on the *ordering type* attribute of the multiple instance activity object. The flow of the tokens in the corresponding GN construct (see Figure 7.b) is as follows:

- A token from the incoming sequence flow place L_I splits to several tokens, which move to L_W . Place L_W contains all tokens, waiting to enter the activity place L_A and perform the activity.
- The tokens move from L_W to L_A either simultaneously or one by one, depending on the type of the activity object: If the ordering is sequential, the capacity of L_A is 1 and the tokens enter it one by one. If the ordering is parallel, the capacity of L_A is infinite and the tokens enter it simultaneously, which means that the activity is performed concurrently.
- When a single token finishes its work in the activity place, it moves to place L_R , where all ready tokens are contained.
- According to its *flow condition* attribute, the multiple instance activity finishes its work when one, several or all of the tokens are ready (i.e. in place L_R). Then all the tokens in L_R move to the output place L_O , where they are merged into a single token, while all other tokens, which can currently be either in L_W or in L_A , exit the GN through places L_{G1} and L_{G2} , respectively.

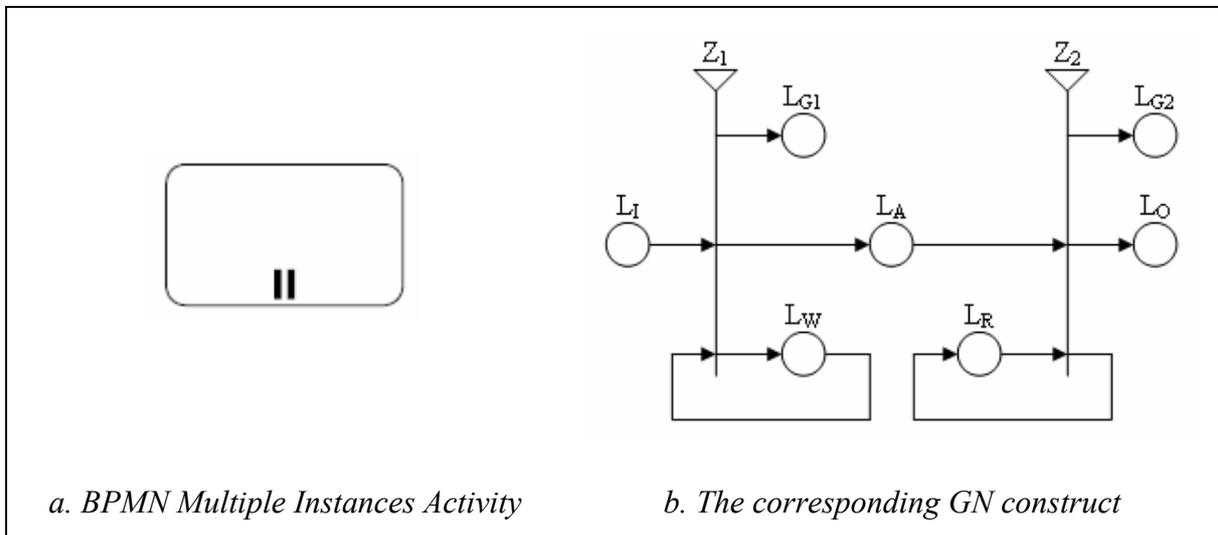


Figure 7: BPMN multiple instances activity and its corresponding GN construct

5 Conclusions

Mapping BPMN to GN gives us a powerful combination of business friendly visual diagramming with the robust execution/simulation capabilities of a formal mathematical model.

This paper shows that:

- GN is powerful enough to represent the complete BPD element set;
- GN representation of BPMN diagrams is fairly simple and straightforward;
- GN and BPMN are similar in scope but have different focus. BPMN focuses on the visual representation while GN on execution and simulation, so they are naturally complementary.

Representing BPD as a GN gives us several benefits as follows:

- The opportunity to use the available GN tools as execution environment;

- We can simulate BPDs in the currently available GN environment, which can help analyzing the behaviour of the business process in case of concurrent process instances;
- We can use GN theory to indirectly analyse BPDs;
- We can define GN operators that will modify the GN-represented BPD (e.g. we can reduce, extend, normalize, etc.);
- We can search for patterns in the GN-presented BPDs.

This paper shows that BPDs can be easily represented with GN. On the other hand it will be quite useful if we could represent GNs using BPMN. This way we could benefit from the visual expressiveness of BPMN when working with GNs and even improve BPD modelling power.

A typical scenario for BPMN to GN and vice versa transformation is:

1. Convert the BPD to GN;
2. Apply GN operators;
3. Convert the GN back to BPD.

An example for such a scenario is the common requirement for most of the BPM diagramming tools: “Show only activities that are of type ‘User’ and hide all the others.” This will allow the user to have a better understanding of the diagram and hence design better diagrams. This feature will improve readability of the BPD even more without losing modelling power.

References

- [1] Business Process Modelling Specification, OMG, January 2009, <http://www.omg.org/spec/BPMN/>
- [2] Atanassov K., On Generalized Nets Theory, Academic Publishing House “Prof. Marin Drinov”, Sofia, Bulgaria, 2007