Transforming Business Patterns to Labelled Petri Nets using Graph Grammars

Karima Mahdi, Allaoua Chaoui, and Raida Elmansouri

MISC laboratory, Department of Computer Science, University Mentouri Constantine, Algeria
mahdi_karima68@yahoo.fr, a_chaoui2001@yahoo.com, raidaelmansouri@yahoo.fr

Abstract:
In this paper we propose an approach and a tool for transforming business patterns to labelled Petri nets for which efficient analysis techniques exist. We specify first, business patterns and labelled Petri nets Meta-Models in UML Class Diagram formalism with the Meta-Modelling tool Atom^3, and then we generate visual modelling tools according to the proposed Meta-Models. Finally, we define a graph grammar which transforms Business Patterns models to Labelled Petri Nets model for analysis purposes. The approach is illustrated with examples.

Key words: Business Patterns, Labelled Petri Nets, Meta-Models, Graph Transformation.

1. Introduction:

To avoid several and dangerous errors in business patterns models, many researchers proposed the mapping of business patterns to Petri net theory [8], which Provide a formal approach to process modelling. However, several patterns are difficult, if not impossible, to realize using this theory. Examples are patterns dealing with multiple instances, and advanced synchronization patterns. In [10] the authors proposed a deterministic Petri net language which implemented the main business patterns proposed in [9]. Our approach achieves this mapping automatically with the Multi-formalism and the Meta-Modelling tool Atom^3 [2].
.
Atom3 was developed at the Modelling, Simulation and Design Lab in the School of Computer Science of McGill University, Written entirely in Python. AToM^3 is a visual tool for meta-modelling and model-transforming. Meta-modelling refers to modelling formalism concepts at a meta-level, and model-transforming refers to automatic converting, translating or modifying a model of a given formalism into another model of the same or different formalism [2].

In this paper, we illustrate how Meta-Modelling is used to design business patterns (BP) and Labelled Petri Nets (LPN) meta-models then to transform BP models to LPN ones.
The remainder of the paper is structured as follows. In section 2, we discuss some basics of BP and LPN. In section 3 we propose a new approach for mapping BP models to LPN ones. In section 4 we apply the proposed approach on a BP model.
Finally, in section 5 we conclude this paper and present some topics for further research.

2. Background

2.1. The Basics of Business Patterns

A BP is a diagram composed of a set of activity nodes, denoting business events; and control nodes capturing the flow of control between activities such as AND-split, AND-join, XOR-split and XOR-join. Activity nodes and control nodes can be connected by means of a flow relation in almost arbitrary ways.

The patterns range from very simple patterns such as sequential routing to complex patterns involving complex synchronizations such as the discriminator pattern. The most relevant patterns can be classified into six categories [9]: Basic control flow patterns, advanced branching and synchronization patterns, Structural patterns, Patterns involving multiple instances, State-based patterns and Cancellation patterns.

It is important to note that the scope of our patterns is limited to static control flow

2.2. The Basics of Labelled Petri nets

A classical Petri net consists of places and transitions connected by arcs, places may contain tokens.
In [1] authors define a deterministic Petri Net language generated by a labelled Petri Net as follow:

$\mathbf{PN} = (\mathbf{N}, \tau, \mu_0, \mathbf{F})$

$\mathbf{N} = (\mathbf{P}, \mathbf{T}, \mathbf{A})$ **is a Petri Net,**

$\mathbf{P}$: **a finite set of places,**

$\mathbf{T}$: **a finite set of transitions,**

$\mathbf{A} \subseteq (\mathbf{PxT}) \cup (\mathbf{TxP})$ **is the flow relation,**

$(\mathbf{P} \cap \mathbf{T}) = \phi,$

$\tau : \mathbf{T} \longrightarrow \Sigma$ **a labeling of T in the alphabet** Σ,

μ_0 **is the initial marking,**

$\mathbf{F}$ **is a set of final markings**

2.3. An Informal mapping of BP into LPN

- Activities can be modelled by transitions
- Control nodes are modelled by places and/or by transitions (depending on the control nodes semantics)
- Activities are instantiated if the transition can fire; this is determined by tokens in LPN.
- The initial state of a BP model can be specified by the initial marking of the corresponding LPN model. A start event signals the start of a BP process. We hereafter put a token in the initial place of the LPN model.

3. THE PROPOSED APPROACH

A meta-model of a given formalism specifies the syntax aspect of the formalism by defining the language constructs and how they are built-up in terms of other constructs. BP and LPN meta-models were created with the UML class diagram formalism of AToM^3.

3.1. **Modelling BP with AToM^3**

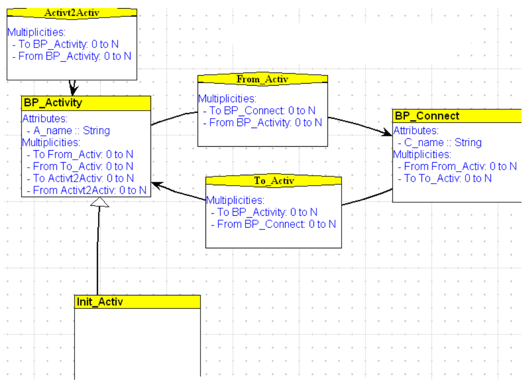


Figure 1: BP meta-Model

The BP Metamodel in figure 1 was constructed in Atom3 according to the BP definition provided above. This metamodel contains three classes and three associations.

- The “BP_Activity” Class designs any business pattern activity; it has only one attributes “A_name” which denotes the name of the activity.

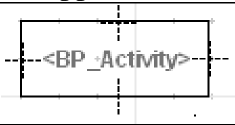
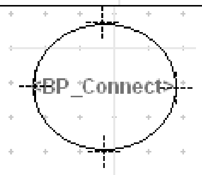
- The “BP_Connect” Class designs control nodes capturing the flow of control between activities such as AND-split, AND-join, XOR-split and XOR-join; it has only one attributes “C_name” which denotes the name of the connector.

- The “Init_Activ” Class designs the initial activity of the BP, it inherit from the “BP_Activity”.

- “From_Activ” and “TO_Activ” are two associations designing the input and output arcs of BP activities.

- The “Activ2Activ” association is used to create sequential activities.

Finally, we define the Appearance property of each construct in accordance with the following notation.

Object	Graphical appearance
Object of “BP_Activity” the class	
Object “BP_Connect” of the class	

When the metamodel is defined, we can generate the BP modelling tool (see figure 2)

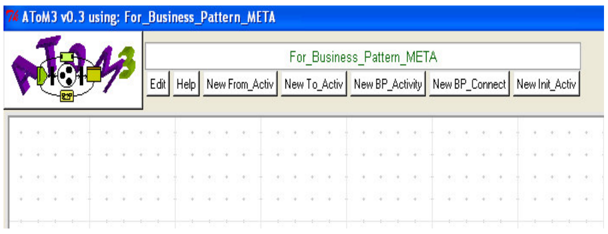


Figure 2: A tool for manipulating BP

3.2. Modelling Labelled Petri nets with AToM^3

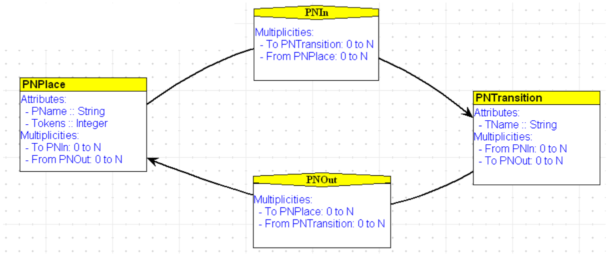


Figure 3. Meta-Model of LPN

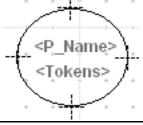
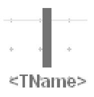
The LPN Meta-model in figure 2 was constructed in Atom3 according to the LPN definition provided above. This metamodel contains two classes and two associations.

- The “PN_Place” Class designs any place in the LPN, it has two attributes the name of the activity “P_name” and the number of tokens in the place “Tokens”.

- The “PN_Transition” Class designs any transition in the LPN; it has one attribute the name of the transition “TName”.

- “PNIn” and “PNOut” are two associations designing the input and output arcs of LPN transitions.

Finally, we define the Appearance property of each construct in accordance with the following notation:

Object	Graphical appearance
Object of “PN_Place” class	
Object of “PN_Transition” class	

When the metamodel is defined, we can generate the BP modelling tool (see figure 4).

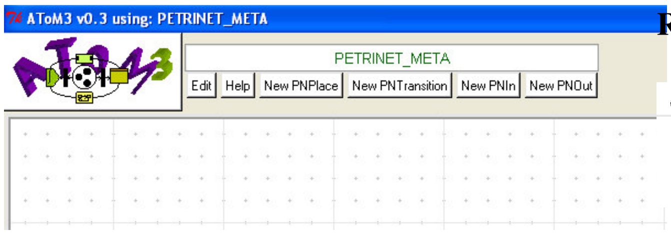


Figure 4: A tool for manapulating LPN

3.3. Transformation of BP to LPN

In Atom3 model transformations are specified through graph grammars, and consist of Initial Action, Final Action and a Set of transformation rules. Each rule defines how to graph rewrite left hand side (LHS) to right hand side (RHS). LHS is a pattern which is matched against model being transformed; RHS is a graph that is inserted into the model instead of a matched Sub graph. The complete definition of a rule consists of: Name, Order, LHS, RHS, Condition and Action.

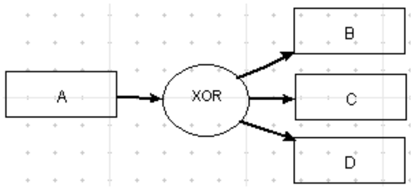


Figure 5: A BP Model example

To transform BP models to LPN ones, we have proposed a grammar with 23 rules. For lack of space, we give in the following the rules that transforms the BP model in figure 5 to the LPN model in figure 6.

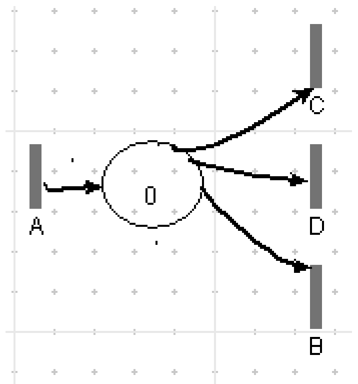


Figure 6: LPN of the BP model of figure 5

Rule 2 (priority 2):

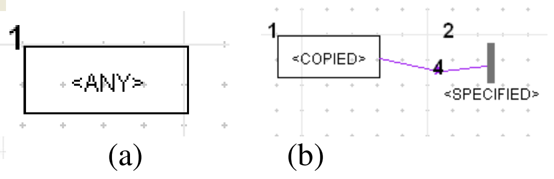


Figure 7: (a) LHS of rule 2 (b) RHS of rule 2

Brief Description: this rule is applied to attach each BP Activity (not previously processed) to a new LPN transition, and specified that the name of the attached transition is the same name of the corresponding BP Activity.

Rule 5 (priority 5):

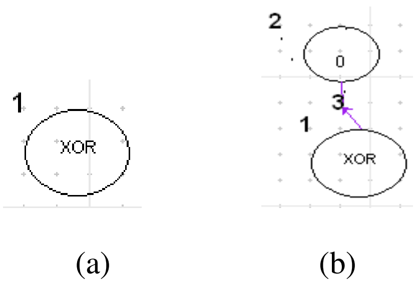


Figure 8: (a) LHS of rule 5 (b) RHS of rule 5

Brief Description: this rule is applied to Attach each BP connector XOR to a new place.

Rule 6 (priority 6):

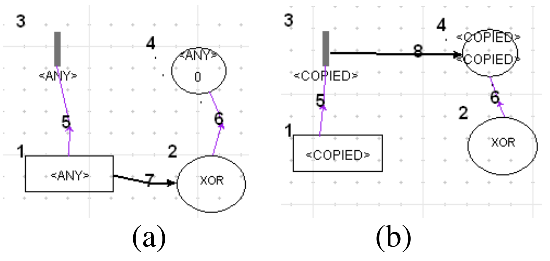


Figure 9: (a) LHS of rule 6 (b) RHS of rule6

Brief Description: this rule is applied to locate an Arc from an Activity to an Xor Link in the model, and then create an Output Arc from the Transition (attached to the Activity) to the Place (attached to

Xor Link). Also, this rule removes the Arc from Activity to Xor Link.

Rule 8 (priority 8):

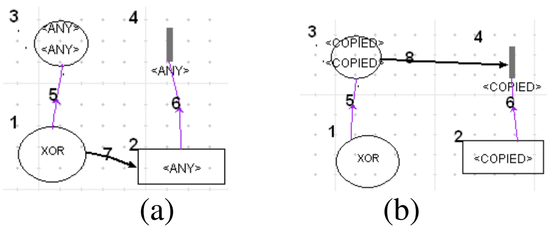


Figure 10: (a) LHS of rule 8 (b) RHS of rule8

Brief Description: this rule is applied to locate an Arc from an Xor link to an Activity in the model, and then create an Output Arc from the Place (attached to Xor Link) to the Transition (attached to the Activity) . Also, this rule removes the Arc from Xor Link to Activity.

Rule 18 (priority 18):

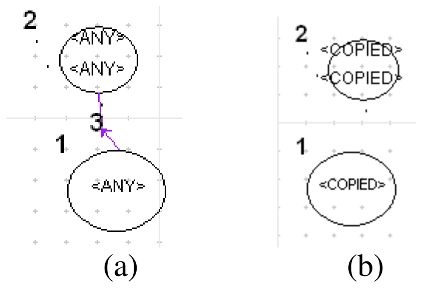


Figure 11: (a) LHS of rule 18 (b) RHS of rule18

Brief Description: this rule is applied to remove generic link between any BP connector and LPN place.

Rule 19 (priority 19):

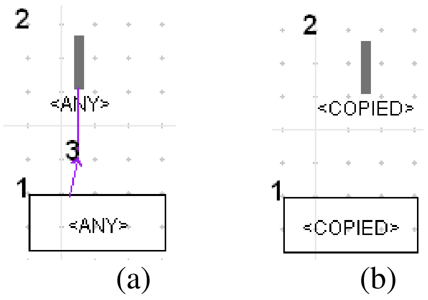


Figure 12: (a) LHS of rule 19 (b) RHS of rule19

Brief Description: this rule is applied to remove generic link between any BP activity and LPN transition.

Rule 20 (priority 20):

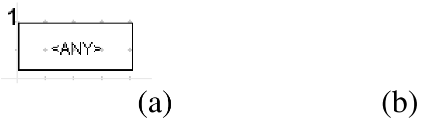


Figure 13: (a)LHS of rule 20 (b) RHS of rule20

Brief Description: this rule is applied to remove all the BP activities

Rule 21 (priority 21):

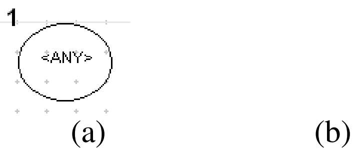


Figure 14: (a) LHS of rule 21 (b) RHS of rule21

Brief Description: this rule is applied to remove all the LPN connectors.

Each rule in the grammar may have condition and action. The following lines give the condition and the action of the rule 2:

Condition

```
node = self.getMatched(graphID,
self.LHS.nodeWithLabel(1))

return not hasattr(node, "_uniqueName8")
and   (node.A_name != 'Start')
```

Action

```
node = self.getMatched(graphID,
self.LHS.nodeWithLabel(1))

node._uniqueName8 = True

pass
```

To transform the BP model in figure 5 to the LPN model in figure 6, ATOM^3 executed the previous rules in this order:

- Rule 2 (4 times)
- Rule 5 (once)
- Rule 6 (once)
- Rule 8 (3 times)
- Rule 18 (once)
- Rule 19(4 times)
- Rule 20(4 times)
- Rule 21 (once)

4. Case Study

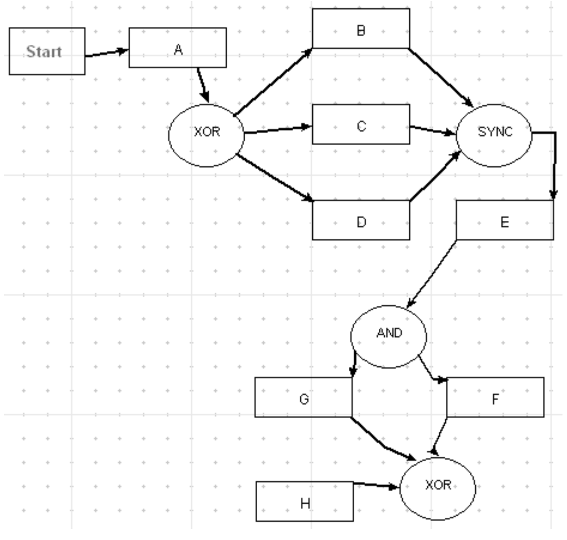


Figure 15: BP Model of the case study

We have applied our tool on the case study of Figure 15 representing a BP model created with our BP modelling tool. It contains the five basic control flow patterns: Sequential pattern, Exclusive Choice pattern(XOR-Split), Simple merge pattern(XOR-Join) , parallel split pattern (AND) and synchronisation pattern(SYNC).
The result of our model-transforming is the LPN model shown in figure 16.

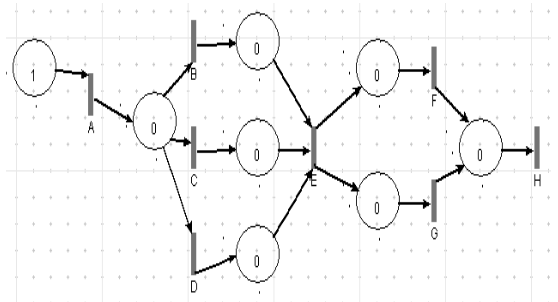


Figure 16 : LPN of the BP model of the case study

5. Related Work

There are many research works in the field of model transformation by graph grammars in the literature. In [3] the authors presented a transformation from statecharts (without hierarchy) to Petri Nets. In [4], we have provided the INA Petri net tool [5] with a graphical environment. First, we have proposed a meta-model for Petri net models and used it in the meta-modeling tool AToM3 to generate automatically a visual modeling tool to process models in INA formalism. Then we defined a graph grammar to translate the models created in the generated tool to a textual description in INA language (INA specification). Then the INA is used to perform the analysis of the resulting INA specification. In [6], we have presented a formal framework (a tool) based on the combined use of meta modeling and Graph Grammars for the specification and the analysis of complex software systems using G-Nets formalism. Our framework allows a developer to draw a G-Nets model and transform it into its equivalent PrT-nets model automatically. To perform the analysis using PROD analyzer, our framework allows a developer to translate automatically each resulting PrT-Nets model into PROD's net description language. To this end, we have defined a meta-model for G-Nets formalism and another for PrT-Nets formalism. Then the meta-modeling tool ATOM^3 is used to automatically generate a visual modeling tool for each formalism according to its proposed meta-model. We have also proposed two graph grammars. The first one performs the transformation of the graphically specified G-Nets models to semantically equivalent PrT-Nets models. The second one translates the resulting PrT-Nets models into PROD's net description language. In [7] we have proposed an approach for transforming UML statechart and collaboration diagrams to Colored Petri nets models. More precisely, we have proposed an automated approach and a tool environment that formally transforms dynamic behaviors of systems described using UML models into their equivalent Colored Petri Nets (CPN) models for analysis purpose.

6. Conclusion and perspectives

In this paper we proposed an approach to automatically transform a BP with basic patterns to the equivalent category of Petri nets called LPN model. The approach is based on graph transformation and ATOM^3 tool.

In a future work we plan to adapt the proposed approach to deal with advanced patterns and to integrate tools for Petri nets verification such as INA tool [5].

REFERENCES:

[1] M. De Backer and M. Snoeck. Deterministic Petri net languages as business process specification language. Dtew research report 0577, K.U.Leuven, 2005.

[2] De Lara, J and Vangheluwe, H (2002). ATOM^3: A Tool for multi-formalism and meta-modeling. LNCS No 2306, 2002

[3] Juan De Lara and Hans Vangheluwe: "Computer aided multi-paradigm modeling to process Petri-nets and statecharts", International Conference on Graph Transformations (ICGT), Lecture Notes in Computer Science, vol. 2505, pp.239-253, Springer-Verlag, Barcelona, Spain, 2002.

[4] Raida El Mansouri, Elhillali Kerkouche, and Allaoua Chaoui: " A Graphical Environment for Petri Nets INA Tool Based on Meta-Modeling and Graph Grammars", Proceedings of World Academy of Science, Engineering and Technology, ISSN 2070-3740, vol. 34, pp.471-475, October 2008.

[5] INA Home page,
http://www2.informatik.huberlin.de/~starke/ina.html

[6] Elhillali Kerkouche and Allaoua Chaoui: "A Formal Framework and a Tool for the Specification and Analysis of G Nets Models Based on Graph Transformation", International Conference on Distributed Computing and Networking
ICDCN'09, LNCS 5408, pp. 206–211, Springer-Verlag Berlin Heidelberg, India, 3-6 January, 2009.

[7] Elhillali Kerkouche, Allaoua Chaoui, El Bay Bourennane, Ouassila Labbani. A UML and Colored Petri Nets Integrated Modeling and Analysis Approach using Graph Transformation. In Journal of Object Technology, vol. 9, no. 4, 2010, pages 25–43. Available at
http://www.jot.fm/contents/issue_2010_07/article2.html

[8] W.M.P. van der Aalst. The Application of Petri Nets to Workow Management. The Journal of Circuits, Systems and Computers, 8(1):21{66, 1998.

[9] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. QUT Technical report, FIT-TR-2002-02, 2002.
[10] W.M.P. van der Aalst1;2 and A.H.M. ter Hofstede2 Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages
In K. Jensen, editor, *Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002)*, volume 560 of *DAIMI*, pages 1-20, Aarhus, Denmark, August 2002. University of Aarhus.