An Automatic Approach to Verify Business Process Models Using INA Petri Nets Analyzer

Elhillali Kerkouche
Computer Science. Deptt.
MISC Laboratory
University of Jijel.
Jijel, Algeria.
elhillalik{at}yahoo.fr

Raida Elmansouri
Computer Science. Deptt.
MISC Laboratory
University of Constantine 2.
Constantine, Algeria.

Allaoua Chaoui
Computer Science. Deptt.
MISC Laboratory
University of Constantine 2.
Constantine, Algeria.

Khaled Khalfaoui
Computer Science. Deptt.
MISC Laboratory
University of Jijel.
Jijel, Algeria.

Abstract—**Business process models describe how a business works. More specifically, they map out how a business accomplishes missions, activities or tasks. The control and the coordination of business processes is made possible by task control constructs that model behaviors like parallel works, decisions, synchronization and repetition. However, the lack of precise semantics for these constructs makes the detection of control flow anomalies and behavioral inconsistencies difficult. The use of formal methods makes such flaws detection possible. Petri Nets provide a powerful formal modeling language based on solid mathematical fundament and provide various analysis techniques through which properties of the Petri Net model can be analyzed. In this paper, we propose an approach and a tool support to facilitate the analysis and the verification of Business process models using Petri Nets formalism. To make the analysis easier, The Petri Nets INA (Integrated Net Analyzer) tool is used. To achieve this goal, we use the Model-Driven Engineering (MDE) approach which is based mainly on Meta-modeling and Model Transformations, and we employ well-known standards and tools under Eclipse to realize the approach. Our approach is illustrated through an example.**

Keywords- Business Process Modeling; Petri Nets; Model-Driven Engineering (MDE); Meta-Modeling; Model Transformations; Eclipse Modeling project.

I. INTRODUCTION

Business process models specify how a business works. They represent how a business carries out given missions, activities, or tasks [1]. A single model shows how a business accomplishes a single task. It would take many process models to fully detail the "hows" of most real world enterprises. A single process can consist of many actors (people, organizations, systems) performing many tasks. In order to accomplish the overall task, the actors must complete specified sub-tasks in a coordinated manner. These sub-tasks can be performed in parallel or sequential. Moreover, they may require repetition of sub-tasks. Most of these processes have decision points where process flow can branch depending on either the condition of the system or the particular process execution. In cooperative processes, actors must pass information. This information transfer can be the trigger for an actor to begin a sub-task. In fact, other triggers are possible, such as time or interrupts. Some processes are ad-hoc. That is, the sub-tasks do not have well defined triggers. Actors may not need to complete all of a subtask before they (or another actor) start working on another dependent subtask. Finally, a process can look differently when described from the viewpoint of different actors [2].

A Business process modeling methodology needs to be able to represent these different aspects of a process description. Business Process Modeling (BPM) provides a conceptual basis for the specification of all business procedures [3]. The control and the coordination of business processes is made possible by task control constructs that model behaviors like synchronization, decisions, parallel works and repetition. However, the lack of firm semantics for these constructs makes the detection of control flow anomalies and behavioral inconsistencies difficult. Formal methods are well suited for the detection of such flaws. Petri Nets provide a powerful formal modeling language based on solid mathematical fundament and provide various analysis techniques through which properties of the Petri Net model such as liveness, reachability and deadlock can be analyzed.

In this paper, we focus on the modeling and analysis issues involved in establishing logical and syntactical correctness of Business process specifications before they are implemented. More precisely, we propose an approach and a tool support to facilitate the modeling and the verification of Business process models using Petri Nets formalism. To make the analysis easier, The Petri Nets INA (Integrated Net Analyzer) [4] tool is used. The work is based on ideas presented in [5], [6], [7] and [3]. In order to achieve our objective, we propose to use the Model-Driven Engineering (MDE) approach which is based on Meta-modeling and Model Transformations, and to employ well-known standards and tools under Eclipse to realize our automatic approach.

The rest of the paper is organized as follows. Section II outlines the major related work. In section III, we present some

concepts of process modeling that are relevant with our work. In section IV, we present the Petri nets formalization of Business process. In section V, we give an overview of the Eclipse Modeling project. In section VI, we propose our approach and apply it on an example in section VII. The last section concludes the paper and gives some perspectives of this work.

II. Related Works

Many researche works have been done on the formalization of Business process. Most approaches are developed to give formal semantics to Business process models using formal methods. They use several formalisms methods like automata as a base model of formal specifications, π-calculus as a mathematical formalism or Petri Nets as a mathematical modeling language. In [8], the author presents a survey of existing proposals for formal verification techniques of Business process models.

Petri-nets [9] offer the advantage of graphical appeal coupled with a rigorous formalism that has found tremendous use in behavior systems and processes that exhibit asynchronism, concurrency, and determinism [10]. Petri nets are especially attractive for formalizing and analyzing business processes for the following reasons [5]: (i) clear and unambiguous description of process logic, (ii) intuitive ease of a self-documenting graphical formalism that retains complete conceptual clarity, and (iii) extensive analysis capabilities. Moreover, Petri nets allow for a study of both (a) structural properties pertaining to the static aspects of the process's definition, and (b) Behavioral properties pertaining to the dynamic aspects of the process observed during its execution [9].

In this paper, we propose an automatic approach for the analysis of Business process models by using INA Petri nets analyzer. More precisely, the proposed approach transforms Business process model into an equivalent Petri Nets model according to the translation schema defined in [5]. For the automatic analysis and verification, the approach translates the obtained Petri Nets model to the input language of INA tool.

III. Business Modeling

Process modeling aims to produce an abstraction of the process that serves as a basis for detailed definition, study, and possible reengineering to eliminate non-value added activities. The process model must allow a clear and transparent understanding of the activities being undertaken, the dependencies among the activities, and the roles (people, machines, information, etc.) necessary for the process. An activity-centered modeling methodology is used for defining process models in the sense that a process is viewed as a sequence of inter-related tasks. The transfer of control between them is determined by logical operations [5]. For the remainder of this work, we will consider a Business process model to be a collection of elements, where an element is either a task or a task control operand that serves to route the flow of control between the tasks. Figure 1 represents an example showing control flow only.

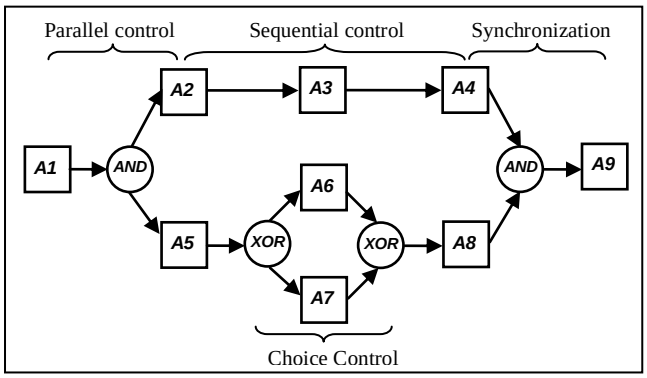

Figure 1. Business process modeling – an example.

We note that the ability for representing and modeling behaviors like concurrency and choice using task control operands increases the chances of defining logically incorrect models with control flow anomalies, the execution of which could result in deadlock, livelock, etc. The focus of this paper is to highlight the use of Petri nets as a technique for formalizing Business process models to analyze verification issues, and to support performance evaluation studies. INA is used to illustrate these issues.

IV. Petri-Net Formalizations of Business Process Models

Any process can be understood to be a collection of events, the conditions that enable these events to occur and the conditions that are satisfied following the completion of these events. Petri Nets ideally describe this intuition. They explicitly separate the conditions and the events involved in a process. The places model the conditions required to enable events which are modeled by the transitions, and state changes are modeled through a simulated movement of tokens.

To map the Business processes to Petri Nets, we have used the ideas proposed in [5]. For example, the Petri Net model in Figure 2 is the mapping of the Business process model in Figure 1.

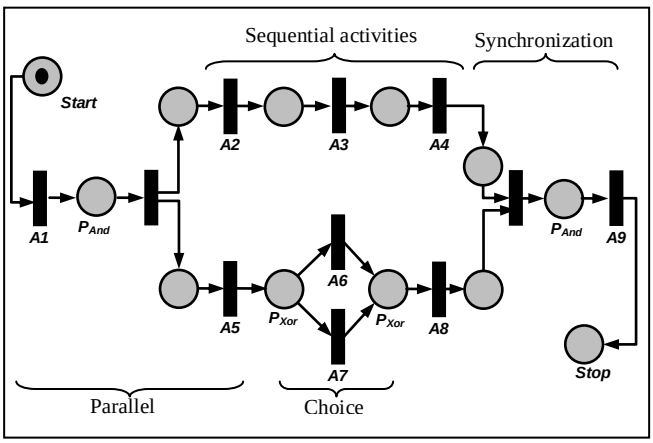

Figure 2. Petri Nets Representation of the Business process model in Figure 1.

V. ECLIPSE MODELLING PROJECT: AN OVERVIEW

The Eclipse Modeling project [11] is a collection of frameworks and tools for model driven Engineering under Eclipse platform. In short, they provide a wide range of solutions for various aspects of model driven development, from language definition to editor construction to code generation as well as model verification and validation.

In the following, we introduce some of the tools from Eclipse Modeling project that have been used in this work.

A. *Eclipse Modeling Framework (EMF)*

The Eclipse Modeling Framework (EMF) [12] forms the basis for all Eclipse Modeling Project tools. It represents the modeling Framework and the code generation facility for specifying meta-models and managing model instances. More precisely, EMF includes its own meta-model called Ecore which is used for defining the abstract syntax of modeling languages. From a modeling language specification defined by Ecore meta-model, EMF generates a simple tree oriented editor that enable viewing and editing instances of the modeling language.

B. *Graphical Editing Framework (GEF)*

The Graphical Editing Framework (GEF) [13] provides technology to aid developers in creating rich graphical editors, which are not easily built using native widgets found in the base Eclipse platform. It contains an entire set of tools to define a graphical concrete syntax for each entity of the meta-model according to its appropriate graphical notation.

C. *Graphical Modeling Framework (GMF)*

The Graphical Modeling Framework (GMF) [14] provides a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF. In other words, it provides a generative bridge between the EMF (that allows the meta-model definition) and GEF (a lightweight graphical framework, based on MVC architecture) to help developers creating enhanced graphical editors.

D. *ATLAS Transformation Language (ATL)*

The ATLAS Transformation Language (ATL) is a model transformation language that allows both declarative and imperative style for transforming definitions [15]. The preferred style of transforming writing is declarative, which means that simple mappings can be expressed easily. However, imperative constructs are provided so that some mappings, too complex to be declaratively handled, can still be specified. An ATL transformation definition is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target model [16]. The source models, the target models and the transformation definition conform to their meta-models as shown in Figure 3.

In addition to model-to-model transformation, ATL uses queries for model to primitive type value transformation. The queries can be seen as operations which calculate values from input models.

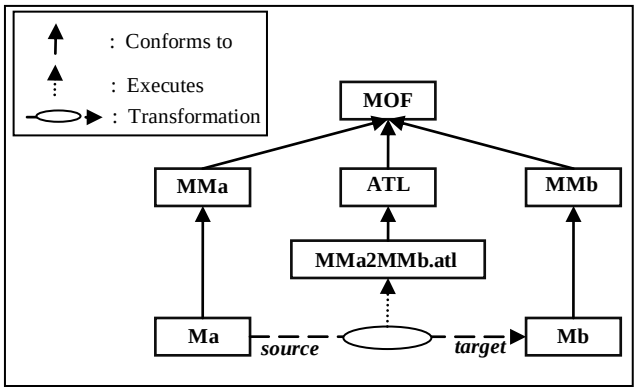

Figure 3. Overview of ATL transformational approach.

In our work, ATL is a means used to specify how to convert Business process model into Petri Nets model, and how to produce INA analyzer code which is a String value from Petri Nets model.

VI. OUR APPROACH

In this section, we describe our automated approach that transforms Business process models into their equivalent Petri Nets models for properties verification using the INA Petri Nets analyzer. The approach is based on the use of well-known standards defined in MDE approach under Eclipse platform. In order to derive the Petri Nets model from Business process specification, we have automated the approach proposed in [5]. To make the analysis easier, we have also automated the generation of the equivalent description of the obtained Petri Nets model in the input language of the INA analyzer (see Figure 4).

Our approach consists of a process with two steps:

The first step consists of Meta-Modeling business process and Petri Nets formalism. Then, we have built graphical editors for both languages according to their proposed Meta-Models.

The second step is to define the model transformations. In order to reach an automatic and correct process of transformation, we have proposed to use ATL transformation language to define and implement the transformations. For this end, we have proposed two model transformations. The first one converts the Business process model to Petri Nets model, whereas the second transformation rebuilds the Petri Nets model in the input language of the INA Petri Nets analyzer tool.

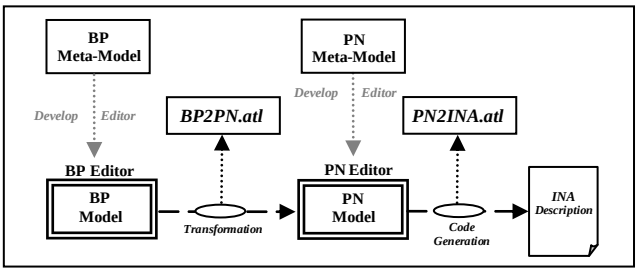

Figure 4. The Proposed Approach .

A. *Meta-Modeling of Business Process and Petri Nets*

To define a modeling language, one has to provide abstract syntax (i.e meta-model denoting constructs, their attributes, relationships and constraints) as well as concrete graphical syntax information (the appearance of constructs and relationships in the graphical editor). In Eclipse EMF, a meta-model is created and defined in the Ecore format, which is basically a sub-set of UML Class diagrams.

Since business processes consist of activities (Tasks) and two kinds of connectors (XOR connector and AND connector) and each task may be linked to a connector by an input arc or an output arc, we have proposed to meta-model business processes with the Ecore model shown in Figure 5.

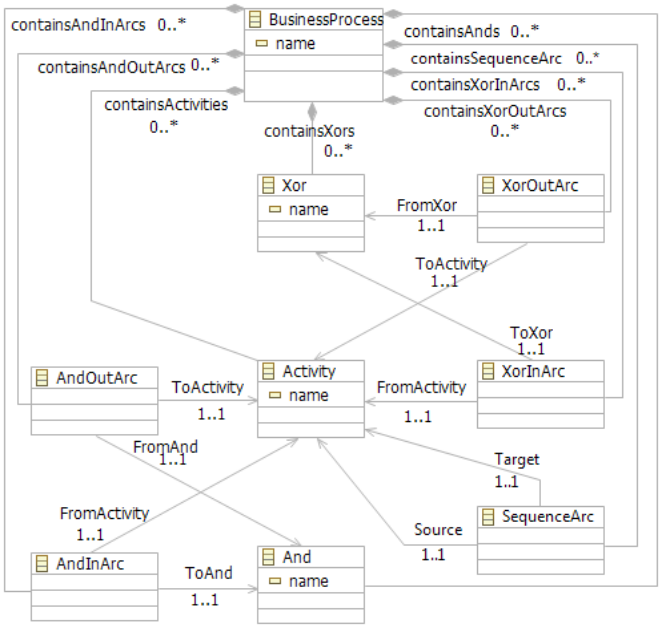

Figure 5. Business Process meta-model in Ecore.

A Petri Nets model is composed of places, transitions, and arcs from places to transitions and from transitions to places. To meta-model Petri Nets, we have proposed the Ecore model shown in Figure 6. The number attached to an arc (*weight* attributes) specifies the number of tokens that are consumed in the source place or produced in the target one. Petri Nets marking is defined by the *numOfToken* attributes of places.

From those proposed Ecore models, we have used EMF to generate a simple tree oriented editor for each one that enables viewing and editing models instances. To develop their graphical modeling editors, we have used GEF and GMF to define the graphical concrete syntax for both languages according to their appropriate graphical notations as shown in Figure 9 and Figure 10.

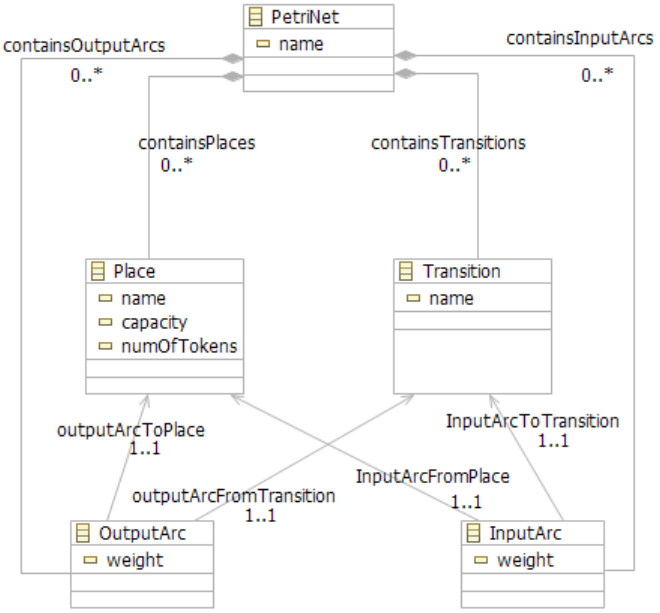

Figure 6. Petri Nets meta-model in Ecore.

B. *Model Transformations*

As we mentioned earlier, we have defined two model transformations in ATL language. Two kinds of transformations are used: model-to-model transformation for transforming Business process model into Petri Nets model, and model-to-text transformations for Rebuilding Petri Nets model in the input language of the INA Petri Nets analyzer tool. The transformation process is achieved by the application of rules. A transformation rule consists in transforming a concept outlined in the source meta-model to a corresponding concept in the target meta-model.

In the following subsections, we describe the rules for these model transformations for our approach.

1st Model Transformation (*BP2PN.atl*): Transforming a Business Process model into a Petri Nets model. To transform the Business processes to Petri Nets, we have used the ideas proposed in [5]. The translation schema is given in section IV. This transformation is defined using seven rules. Figure 7 shows some representative rules. The first translates a Business Process activity into a Petri Nets transition. The second rule converts an *Xor* link into Petri Nets place. The last rule considers the whole Business Process model and builds the associated Petri Nets model.

2nd Model Transformation (*PN2INA.atl*): Generating the equivalent INA description of the resulted Petri Nets model. In order to manipulate the obtained Petri Nets model inside INA analyzer tool, we have composed the preceding transformation with a query *PN2INA* (see Figure8) that translates the Petri Nets model into a textual form (.pnt) conforming to the textual syntax of the INA tool.

```
module BP2PN;
create OUT : PetriNet from IN : BusinessProcess;

--- Transformation rules
rule Activity2Transition{-- rule N°01: transforms
activities into their equivalent transitions.
         from actv :BusinessProcess!Activity
         to
              trans:PetriNet!Transition
                   (
                  name<-actv.name
                        )
}
rule XorLink2Place{-- rule N°02: transforms XorLinks into
their equivalent Places.
         from  Xor_Link :BusinessProcess!Xor
         to
              P_ForXor:PetriNet!Place(
                  name<-Xor_Link.name
                  , numOfTokens <- 0
                  )
}
rule SequenceArc2Place{ -- rule N°03 …
rule XorInArc2Trans_OutArc{-- rule N°04 …
rule XorOutArc2Trans_InArc{-- rule N°05 …
rule AndOutArc2Trans_InArc{ -- rule N°06 …
rule BusinessProcess2PetriNet{-- rule N°07: builds the
equivalent PN model
         from BP:BusinessProcess!BusinessProcess
         to
              PN:PetriNet!PetriNet(
                  name<-BP.name
                     , containsPlaces<-
PetriNet!Place.allInstances()
                  , containsTransitions<-
PetriNet!Transition.allInstances()
                  , containsInputArcs<-
PetriNet!InputArc.allInstances()
                  , containsOutputArcs<-
PetriNet!OutputArc.allInstances()
                  )
}
```

Figure 7. 1st MT: Somme ATL Rules.

```
query PN2INA =
         Default!PetriNet.allInstances()->asSequence()->
         first().generatePN().writeTo('C:/result_PN.pnt');
helper
context Default!PetriNet
def : generatePN() : String =
    'P   M  PRE,POST  NETZ 1:' +
    self.name.toString()+'\r\n'
    + Default!Place.allInstances()->iterate(a;acc :
    String=''|acc + a.generateNetStructure().toString()) +
    '@' + '\r\n' +
    'place nr.          name capacity time'+'\r\n' +

    Default!Place.allInstances()->iterate(a;acc :
    String=''|acc + a. generatePlace().toString()) +

    '@' + '\r\n' +
    'trans nr.          name priority time'+'\r\n' +
    Default!Transition.allInstances()->iterate(b;acc :
    String=''|acc + b. generateTransition().toString()) +
    '@ '
    ;
helper
context Default!Place
def : generateNetStructure() : String = …

helper
context Default!Place
def : generatePlace() : String = …

helper
context Default!Transition
def : generateTransition() : String = …
```

Figure 8. 2nd MT: ATL query PN2INA.

VII. CASE STUDY

To evaluate the practical usefulness of proposed approach, we consider a simple example of Business process model which represents a deadlock situation. Figure 9 presents the model created in our editor.

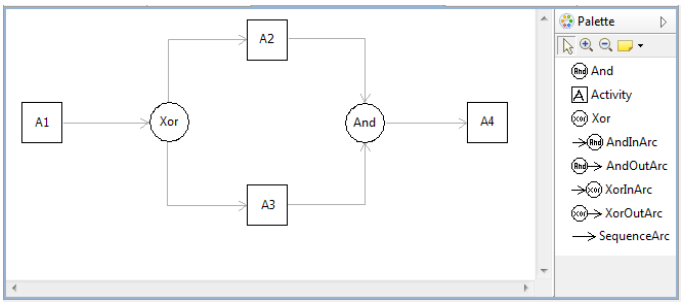

Figure 9. An example of Business process model.

In order to analyze this Business process model, we have to transform this specification into its equivalent Petri Nets model. To realize this transformation in our approach, we have to execute the *BP2PN.atl*. The resulted Petri Nets model of the automatic transformation is shown in Figure 10. We have added two places: *Start* place with one token and *Stop* place.

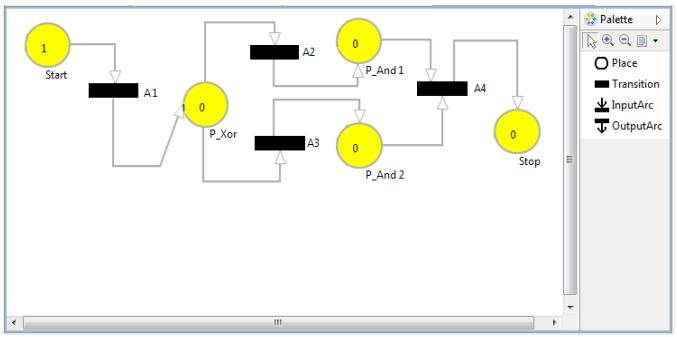

Figure 10. Resulted Petri Nets model.

In order to perform the analysis of the resulted Petri Nets model using the INA analyzer, we have to generate its equivalent INA description. To generate INA description in our approach, we have to execute the *PN2INA.alt* defined in the previous section. The automatic generated file which contains the INA description is shown in Figure11.

To verify the properties of the model, we have invoked the INA tool with the generated INA specification file as input. Then, the INA tool provides the properties of the Petri Net as shown in Figure 12.

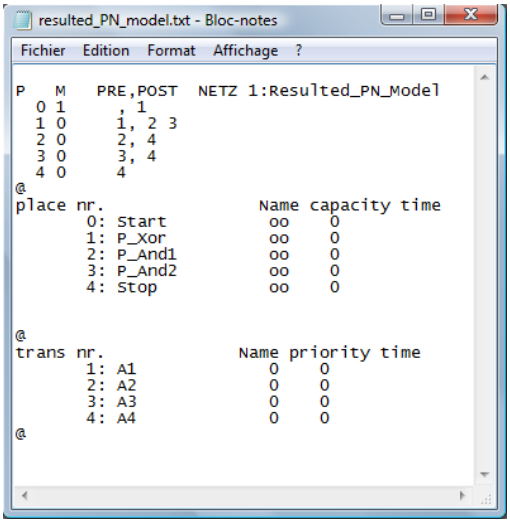

Figure 11. The Generated INA specification.

We can see from INA screen that the Net is not live, not safe and the deadlock-trap property is not valid. So, there is a deadlock situation.

We have also used our approach to verify the situations of multiple repetition and livelock and we have obtained the expected results.

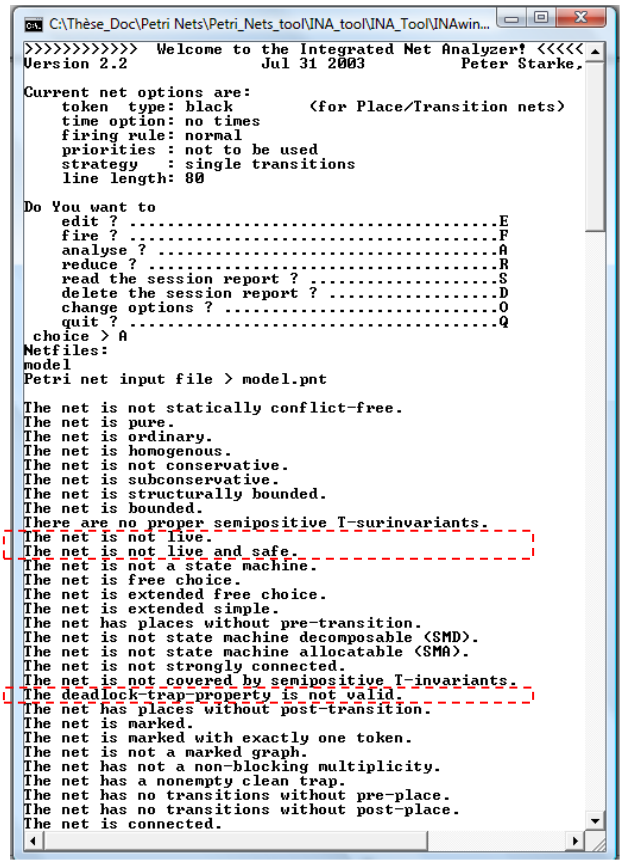

Figure 12. Verification of the obtained Petri Nets model.

VIII. CONCLUSION

In this paper, we have reported the use of Model-Driven Engineering principles for automatic verification of Business process models using INA Petri Nets analyzer. More precisely, we have proposed an automated approach that transforms Business process models into their equivalent Petri Nets models for analysis purposes. These transformations aimed to bridge the gap between informal notation (Business process models) and more formal notation (Petri Nets). It produces graphical and rigorously-analyzable models that facilitate early detection of anomalies. To make the analysis easier, we have used the obtained Petri Nets models to generate automatically their equivalent description in the input language of the INA Petri net analyzer. The proposed approach is developed under Eclipse and implemented using Eclipse Modelling Project technologies. The Business process models and Petri Nets are defined using Ecore models, whereas the transformation process is defined and executed using ATL language. In a future work, we plan to back-annotate the verification results into the Business process model to reach the complete automation of the transformation.

REFERENCES

[1] B. Curtis, M. I. Kellner, and J. Over, "Process Modeling". Comm. of the ACM, 35(9):75–90, 1992.

[2] D. Georgakopoulos,M. Hornick, and A. Sheth,"An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure". Distributed and Parallel Databases, 3:119-153, 1995.

[3] W. M. P. Van der Aalst, A. H. M. ter Hofstede, and M. Weske, "Business Process Management: A Survey", Lecture Notes in Computer Science 2678 Springer, ISBN 3-540-40318-3 2003.

[4] INA Home page, http://www2.informatik.hu-berlin.de/~starke/ina.html

[5] E. Sivaramanm and M.Kamath, "On The Use of Petri Nets for Business Process Modeling", Proceeding of the 11th Annual Industrial Engineering Research Conference, Orlando, FL. 2002.

[6] W. M. P. Van der Aalst, "Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques". In Aalst, W.M.P., Desel, J., and Oberweis, A., editors, Business Process Management – Models, Techniques, and Empirical Studies, volume 1806 of Lecture Notes in Computer Science, pages 161–183. Springer-Verlag, 2000.

[7] R. El Mansouri, "On the use of Meta-Modelling and Graph Grammars to Generate Petri Nets models for Business Processes", IRECOS Journal, January, 2008 issue.

[8] S. Morimoto, "A Survey of Formal Verification for Business Process Modeling". In : ICCS 2008, pp. 514-522, 2008.

[9] T. Murata, "Petri Nets: Properties, Analysis and Applications", Proc. IEEE, Vol. 77, pp. 541-580, No.4, 1989.

[10] H.J. Genrich and K. Lautenbach. "System Modelling with High-Level Petri Nets ". Theoretical Computer Science, Vol. 13, pp. 109-136, 1981.

[11] EMP Home page, http://www.eclipse.org/modeling/

[12] EMF Home page, http://www.eclipse.org/emf/

[13] GEF Home page, http://www.eclipse.org/gef/

[14] GMF Home page, http://www.eclipse.org/gmf/

[15] F. Jouault and I. Kurtev, "On the architectural alignment of ATL and QVTs", Proceedings of the ACM Symposium on Applied Computing (SAC'06), Dijon, France, April 23-27. 2006.

[16] F. Jouault, F. Allilaire, J. Bézivin and I. Kurtev, "ATL: A model transformation tool", Science of Computer Programming, Elsevier, vol. 72, no. 1–2, ,pp. 31-39 (2008).