Published in : “New Technologies for Information Systems”, Proceedings of the 1st International Workshop on Validation and Verification of software for Enterprise Information Systems, VVEIS 2003, p 94-102, ISBN: 972-98816-2-6

ArgoPN: a CASE Tool Merging UML and Petri Nets

Jérôme Delatour[1] and Florent de Lamotte[1]

ESEO, Ecole Superieure d'Electronique de l'Ouest,
4, rue Merlet de la Boulaye, BP 926, 49009 Angers cedex 01
{jerome.delatour, florent.lamotte}@eseo.fr

Abstract. Although several researchers are developing approaches merging UML with Petri net in order to verify and validate the dynamic representations, there is still a lack of UML CASE tools implementing these approaches. Consequently, industrials are facing difficulties to use and benefit from that kind of approaches. Therefore, we started to develop a free open-source UML CASE tool that would be able to exchange data with existing Petri Net analysis tools. Rather than developing a whole system, we chose the free open-source CASE tool ArgoUML and plugged our own Petri Net editor as a module. Through a pedagogic modeling example, this paper presents the current features and status of this tool, called ArgoPN. Even if ArgoPN is far from being achieved, it is currently integrated and efficient in ArgoUML; Petri Nets can now be used in replacement of the Statechart diagram and have access to UML elements (attributes, methods...). We now expect that Petri Net researchers in UML will consider ArgoPN mature enough to be usable and expandable for their own work.

Keywords: Software engeneering, UML, Verification, Dynamic model, Petri Net, CASE Tool

1 Introduction

The Unified Modeling Language (UML) is becoming widely adopted for the design of many varieties of systems from small control systems to large and complex open systems. Despite this fact, and as it has been recognized by its author, the current UML dynamic semantics is imprecise. Many problems (ambiguity, inconsistency and incompleteness) have been identified in its semantics [1] [2]. Thus, the UML capability for verification and validation is very limited. Unfortunately, large sets of systems need such activities, especially in systems in which the dynamic aspect is important. This problem of providing formal semantics to UML dynamic models has been tackled in different ways [3] [4] [5] [6] [7] ...

A possible technique is to couple UML and Petri Nets (PN). Numerous academic works were conducted in that way, targeting different kinds of systems, from Workflow [8] to real-time systems [9], through communication protocols [10], dependable applications [11], mobile agents [12] or user interface prototyping [13].

Although all of these approaches use UML to describe static aspects and use PN as a supplement to model behavioral aspects, they differ by several criteria:

- The class of PN used (from place/transition net to stochastic [10] through high level timed PN [14] or object PN [15]);

- The way they translate (automatically or manually) UML elements into PN elements (mainly from Interaction Diagrams and Statecharts Diagrams, but also from Use Case Diagrams);
- The type of verifications and validations offered (simulation, executive models, performance or reachability analysis, incoherency detection...).

Though these academic approaches proved their benefits on academic or real case studies, their use in an industrial context is still unusual. One reason is certainly the lack of UML CASE tools hosting a PN support.

For this reason, we started to develop such a tool. Rather than defining a specific framework for a PN/UML approach, we chose to study existing approaches, looking for common principles and implementing a generic platform capable of hosting all of them.

This paper is organized as follow. We will first discuss in section 2 the requirement for this generic UML PN CASE tool. In section 3 we'll sketch our implementation strategy. Section 4 presents, through an example, the current status of our tool, ArgoPN. Finally, section 5 concludes by indicating the ongoing work.

2 Requirements for a versatile Petri Nets UML CASE tool

Our aims were to provide an usable tool, both for academic and industrial communities. On one hand, in order to encourage commercial CASE tools to integrate PN, this tool should be mature enough to demonstrate its feasibility and industrial advantages. On the other hand, it should be generic enough to allow the research community to quickly test their proposals.

As far as the industrial community is concerned, this tool should offer:

- An UML model based on latest specification
- Support for commonly used interchange formats (XMI and XML variants)
- A practical way to read the PN analysis result (UML Diagrams for instance)
- User assistance during the design of UML and PN models.

As far as the researchers community is concerned, the underlying models (PN and UML) need to be strong enough to enable model transformation between UML and PN. Moreover, in order to allow the model extension and prototypes use, it seems important for the tool to be easily extended, either by adding code (through modules) or by using external tools.

3 Defining an implementation strategy

The choices made to fulfill the previously described needs are presented here after.

Choosing an existing UML CASE Tool

We would have spent a long time developing a full UML CASE tool; therefore, we decided to base our studies on an existing one and to devote our work entirely to PN. There are already some UML case tools provided by the open source community like Alma[1], KUML[2], Dia[3], Umbrello[4] and ArgoUML[5].

ArgoUML [16] appeared to be the most achieved tool, fulfilling with all our needs and providing a plug-in mechanism that allows us to write our own PN plug-in without changing anything in its source code. Moreover, one of its main goals is to provide a tool that increases the user's productivity and helps him to master UML. Consequently, ArgoUML offers mechanisms that continuously check the design, report errors, and suggest possible improvements. These critics and wizards will be interesting to guide the modeler in its UML/Petri Nets design.

Pairing UML and Petri Nets description in ArgoUML

In order to handle a PN element in ArgoUML, we need to interact with its meta-model. This meta-model is based on the NSUML[6] library, which is a set of classes generated from the official UML meta-model description. This gives ArgoUML an extreme flexibility, making it virtually possible to switch from a version of UML to another just by regenerating the NSUML classes.

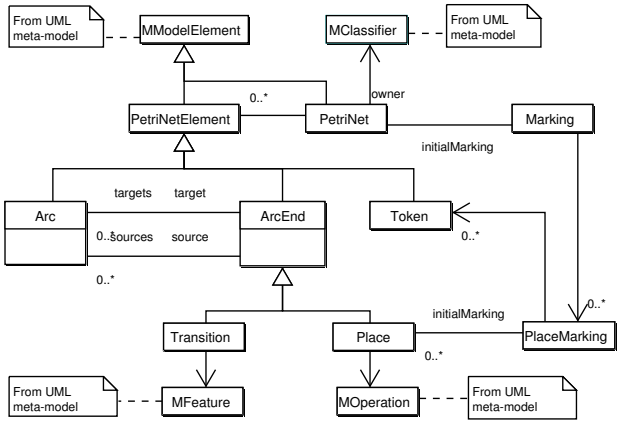

Fig. 1. Our PN meta-model.

[1] http://www.memoire.com/guillaume-desnoix/alma
[2] http://www.informatik.fh-hamburg.de/ kuml/
[3] http://www.lysator.liu.se/ alla/dia/
[4] http://uml.sourceforge.net
[5] http://argouml.tigris.org/
[6] http://nsuml.sourceforge.net

We chose to join the PN and UML models by defining a PN meta-model inside the ArgoUML meta-model, inheriting from the top classes of NSUML; thus, ArgoUML handles PN Elements as any other UML elements.

Our meta-model (as drawn on figure 1) is based on the visual definition of PN instead of the algebraic one, this has two main purposes:

- it enables the extension of the meta-model, in particular into High Level Petri Nets which are easily described this way;
- it will permit transformations between UML and PN models.

You can also notice that we model the marking of the PN. The marking gives the state of the PN. This state can also be considered as the one for the described object. We will use the notion of marking for instanciations of objects described with a PN (each instance will have a distinct Marking).

The user handling of the model

ArgoUML provides three main views to handle the UML model: a tree view to navigate in the model, a diagram editor to edit UML Diagrams and property panels to edit elements in the model.

Of course, a PN Diagram was developed to add PN elements in the model and to define their interactions, property panels were also developed for every PN element.

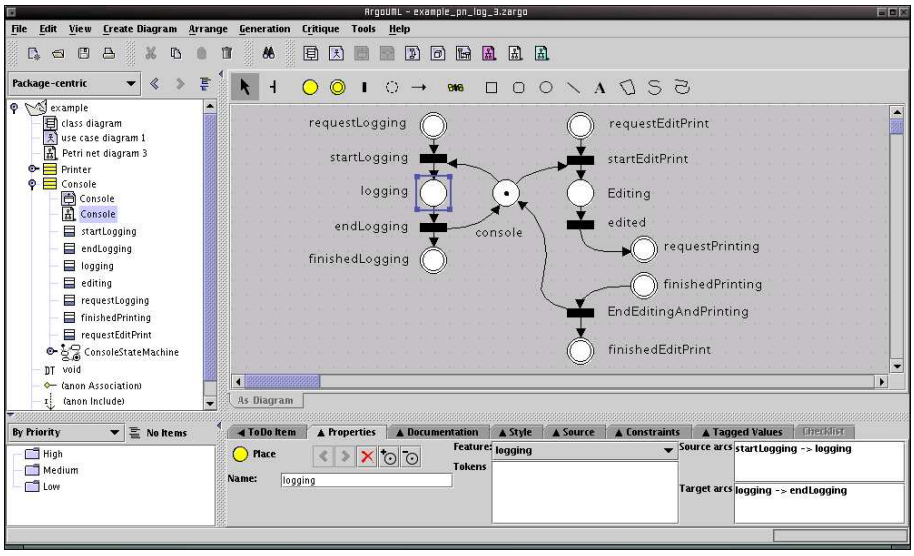

Fig. 2. ArgoPN main window.

Figure 2 presents ArgoPN main window while editing a PN owned by the ”Console” class. As depicted on Figure 2, the selected place on the diagram is linked with the ”logging” operation of the class ”Console”.

Saving the model

ArgoUML saves the UML model in the XMI file format that has been defined as an interchange format between UML CASE tools.

The PN, edited with ArgoUML are saved into the PNML file format [17], an XML interchange format for PN. This format is generic enough to be converted into other XML PN tools format. We have already achieved a support for the PN file format of the Romeo framework[7]. The PN can be saved along with the UML model.

A way to extend the tool

Extending ArgoPN can be done through the Plug-in mechanism provided by ArgoUML. Thus, one can use and modify every resource in the model or the UI. The PN model was designed in a way that makes it easy to support a new PN class.

Extending the tool in such a way should allow anybody to add functionalities not yet offered by ArgoPN, from verification to model transformation or code generation. New PN classes should also be added easily this way.

4 ArgoPN through an academic example

This section presents ArgoPN and its suitability as a modeling tool through a simple example.

The following example (inspired by Robert Valette's slides [8]) shows the interest of modeling a system using PN along with UML. It will first be described by "conventional" UML Diagrams, then with PN Diagrams.

This system is made up of a printer and a console (represented on the Class Diagram Figure 3).

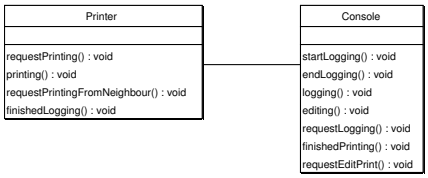

Fig. 3. The Class Diagram for the system.

The console allows a user to edit and then print a text. Moreover, it provides logging facilities to the local printer. Its Statechart Diagram is provided on Figure 4. Please note that one cannot edit while a printer is logging.

The printer can accept documents from local or remote consoles. Each distant printing request will be logged on the local console (see Figure 5).

[7] http://www.irccyn.ec-nantes.fr/irccyn/Equipes /Temps_Reel/romeo

[8] http://www.laas.fr/ robert/enseignement.d/uml_petri.pdf

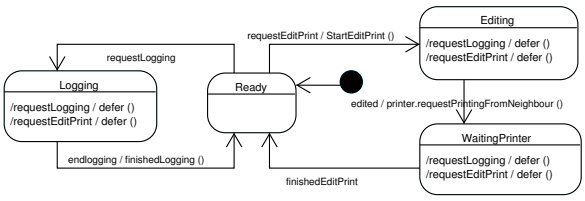

Fig. 4. The Console Statechart Diagram.

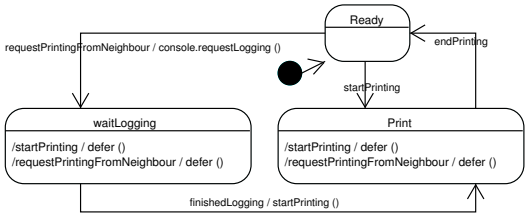

Fig. 5. The Printer Statechart Diagram.

In Figures 6, 7, the PN behaviors of the two classes are described. Here, we chose the most used semantics for coupled with Object Oriented modeling [18]. PN Transitions correspond to events, pre and post conditions, or run-time constraints. Places represent activities, waiting states and operation linking. The current marking of the PN gives the state of the object. The object communications are based on method invocations and are modeled on the PN by a place, shared between the communicating objects. In order to differentiate these shared places from normal ones, they are drawn with two concentric circles

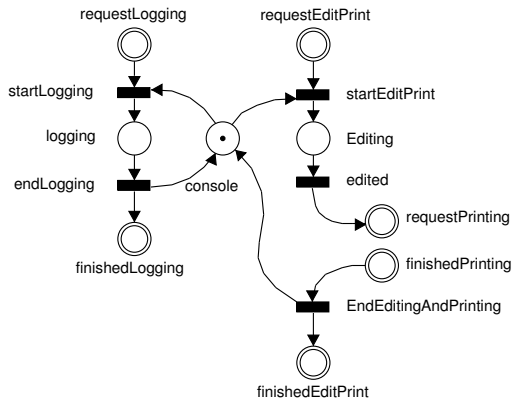

Fig. 6. The PN Diagram for the Console.

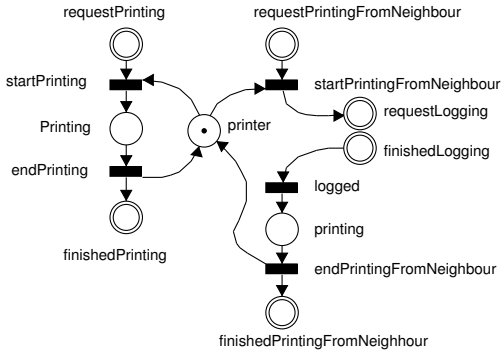

Fig. 7. The PN Diagram for the Printer.

Comparing the PN representation (Figures 6, 7) with the Statechart ones (Figures 4, 5) leads to several statements. In PN representations, the critical resources (the printer and the console) are explicitly represented by a place, whereas on the Statecharts they are implicitly present with the ready state. Execution threads are also more visible on the PN.

When merging the shared communication places of the two PN, the resulting one (see Figure 8) shows immediately a possible deadlock. It occurs when a user wants to print a document from a neighboring office while someone else is editing a document on the local console.

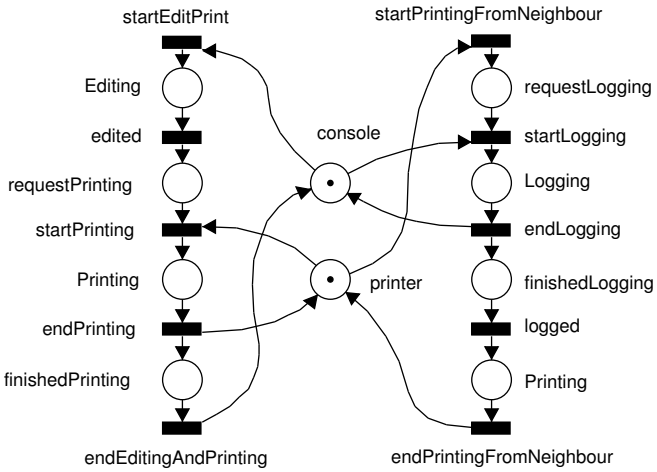

Fig. 8. The global PN of the system.

Of course, PN does not only help to find deadlocks. Various kinds of analysis could be performed, but their presentation is out of the scope of this paper. Our intention is simply to show some advantages of using PN and current ArgoPN possibilities.

5 Perspectives and Conclusion

ArgoPN is currently a PN editor supporting Time and Timed Predicate Transition PN classes linked with the UML meta-model. PN can replace the UML Statecharts Diagram. It can save these PN with the UML model, using PNML and XMI.

As far as our ongoing work is concerned, we scheduled two complementary developments:

- Feedbacks from external analysis tools (using linear logic [19], State Classes Graphs...). The idea is to generate UML Diagrams from the analysis results provided by external PN tools.
- Automated or semi-automated generation of PN from Statecharts and Sequence Diagrams. As these transformations sometimes need guidelines from user, we are now testing the ArgoUML's critics system.

In this paper, we presented a new tool that will hopefully help people needing PN along with an UML model. This project is free software, published under a BSD-like license and available at http://argopno.tigris.org/. In this context, we welcome any developer interested in extending or in using our work.

References

1. Reggio, G., Wierenga, R.: Thirty one problems in the semantics of uml 1.3 dynamics. In: Workshop "Rigourous Modeling and Analysis of the UML Challenges and Limitations" in OOPSLA'99, Denver, Colorado (1999)
2. Kozlenkov, A., Zisman, A.: Checking behavioural inconsistencies in uml specifications, scenarios and state machines: Models, algorithms, and tools. In: ICSE Workshop - ICSE 2002, Orlando (2002)
3. G. Engels, e.a.: Testing the consistensy of dynamic uml diagrams. In: IDPT-02. (2002)
4. Geisler, R.: Precise uml semantics through formal metamodeling. In L. Andrade, et al., E., ed.: Proceedings of the OOPSLA'98 Workshop on Formalizing UML. Why? How? (1998)
5. Lano, K., Bicarregui, J.: Formalising the uml in structured temporal theories. In Kilov, H., B. Rumpe, E., eds.: Proceedings Second ECOOP Workshop on Precise Behavioral Semantics (with an Emphasis on OO Business Specifications), Technische Universitt Mnchen, TUM-I9813 (1998) 105–121
6. Latella, D., Majzik, I., Massink, M.: Towards a formal operational semantics of uml statechart diagrams. In Fantechi, A., Gorrieri, R., eds.: Proc. 3rd IFIP Workshop on Formal Methods for Open Object-based Distributed Systems (FMOODS). (1999) 331–348
7. Fernández, J., Toval, A.: Can intuition become rigorous? foundations for uml model verification tools. In Press, I., ed.: International Symposium on Software Reliability Engineering, San Jose, California, USA (2000)
8. Wirtz, G., Giese, H.: Using uml and object-coordination nets for workflow specification. In: Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'2000). Volume 4., IEEE (2000) 3159–3164

9. Delatour, J., Paludetto, M.: Uml/pno: a way to merge uml and petri net objects for the analysis of real-time systems. In: ECOOP'98, W19, Bruxelles (1998)
10. Pooley, R., King, P.: The unified modeling language and performance engineering. In: IEEE Proceedings-Software. Volume 146, 2. (1999)
11. A. Bondavalli, I.M., Mura, I.: Automated dependability analysis of uml designs. In: 2nd IEEE ISORC'99, Saint-Malo, France (1999) 139–144
12. J. Merseguer, J.C., Mena, E.: Performance analysis of internet based software retrieval systems using petri nets. In: 4th ACM international workshop on Modelling, analysis and simulation of wireless and mobile systems, Rome, Italy (2001)
13. Elkoutbi, M., Keller, R.: User interface prototyping based on uml scenarios and high-level petri nets. In: 21st International Conference on Application and Theory of Petri Nets 2000. Volume 1825 of LNCS., Aarhus, Denmark, Springer-Verlag (2000)
14. Baresi, L., Pezzé, M.: On formalizing uml with high-level petri nets. In: Concurrent Object-Oriented Programming and Petri Nets. Volume 2001. LNCS Springer (2001) 276–304
15. Saldhana, J., Shatz, S.: Uml diagrams to object petri net models: An approach for modeling and analysis. In: International Conference on Software Engineering and Knowledge Engineering, Chicago, Illinois (2000)
16. Robbins, J.E., Redmiles: Cognitive support, uml adherence, and xmi interchange in argo/uml. In: D. F. Journal of Information and Software Technology, Special issue: The Best of COSET'99. Volume 42,2. (2000) 79–89
17. Michael Weber, E.K.: The petri net markup language. In: to appear in Petri Net Technology for Communication Based Systems, Advances in Petri Nets. Volume 2472. LNCS (2002)
18. Paludetto, M., Delatour, J.: Uml et les rseaux de petri: Vers une sémantique des modèles dynamiques et une méthodologie de développement des systèmes temps réel. Revue l'objet **5** (1999) 443–467
19. Taïani, F., Paludetto, M., Delatour, J.: Composing real-time objects: A case for petri nets and girard's linear logic. In: The 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing ISORC 2001, Magdeburg, Germany (2001)