

A Meta-modeling Approach for Sequence Diagrams to Petri Nets Transformation within the requirements validation process

Adel Ouardani
Philippe Esteban
Mario Paludetto
Jean-Claude Pascal

Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS
Université Paul Sabatier
7 Av. du Colonel Roche,
31077 Toulouse Cedex 4, France.
{ouardani, esteban, paludetto, jcp } @laas.fr

KEYWORDS

Meta-modeling, Petri nets, Sequence diagrams, Model transformation, Validation and Verification

ABSTRACT

This paper deals with the transformation of UML Sequence Diagrams into Petri Nets. The involved SD are those joined to the Use Cases dynamic description. In other words, they concern the requirements modeling. The approach is seen in a more general context of heterogeneous system design. It endeavors to integrate, early in the requirement modeling process, a formal model within a semi-formal one, in this case Petri nets with UML SD. The SD to PNs transformation is meta-modeling oriented within the MDA approach, specifically for PIM design. So, a meta-model for SD and one for PNs are first defined. Then, the meta-model of the transformation is done, and the transformation rules may be deduced. As a result, the approach allows a partial automation of the transformation, and subsequently the verification and validation of the system requirements.

1. INTRODUCTION

The Model Driven Architecture “MDA” proposed by the Object Management Group (OMG) is increasingly used to define an approach to software development based on modeling, and automated mapping of models to implementations. It becomes even so for the design of heterogeneous systems too, where several domains and trades are strongly involved. Its recent industrial success, along with its use in an ever greater number of industrial projects has prompted engineers and researchers to define and use MDA based approaches. The basic MDA concept involves defining a Platform Independent Model “PIM” and its automated mapping to one or more Platform-Specific models “PSM” from the Platform Dependent Model too (Frankel 2003). We defined a platform in accordance with such a concept for the virtual prototyping of the system to design. The virtual prototyping designed with an MDA view allows us to make a system V&V (Validation and Verification) by means of both, the formal verification and the system simulation. The aim is to tackle the design and implementation

steps with requirements models free of faults (Kleppe et al. 2003).

The work presented in this paper can be located at the highest level of the PIM development when the customer requirements are translated into models. At the beginning, the PIM production is mainly based on the UML language. The today design trend entails to integrate or extend formal models early in the modeling process because of many benefits to do so, consistency check of the requirements models, system V&V or just to match with some PDM. In our approach the formal model is PNs because our PDM is also PNs based, and also we have a good expertise in using PNs in several ways (Esteban et al. 2005), the formal V&V, the operating of its structural properties as an aid for system structuring and the system simulation that results of the PNs transformation into VHDL-AMS (Albert et al. 2005) (see Fig. 1).

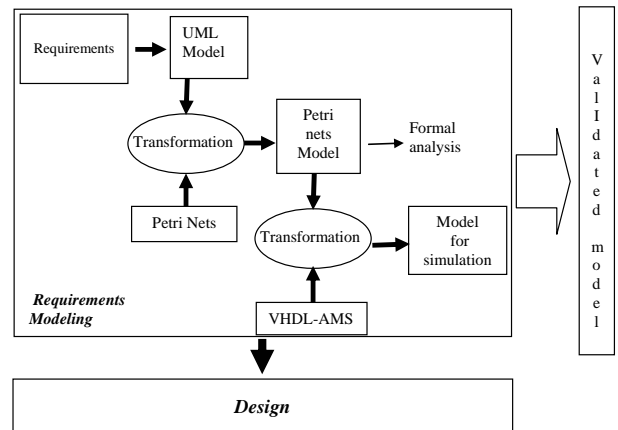


Fig.1. Requirements validation process

While the current OMG standards such as the Meta Object Facility “MOF” and the UML provide a well-established foundation for defining PIMs and PSMs, no such well-established foundation exists for transforming UML PIMs into formal model for V&V purpose (Gerber et al. 2002), (Czarnecki and Helsen 2003) nor into PSMs. In our work the PIMs are related to the requirements modeling, so in this paper only the sequence diagrams associated to the sys-

tem Use Cases are considered. Briefly presented, our PDM mainly comprises Petri Nets for the discrete domain of the system and functional blocks for the continuous one. So, we needed to transform very early the UML models into PN model for two essential goals:

- Formal verification of PIMs based on the properties analysis of the PNs models
- Transformation of PIMS into PSMs with the aim to complete the formal verification by means of the simulation. The simulation is based on a VHDL-AMS heart. So, another model transformation, PNs into VHDL-AMS, has been worked out (Albert et al. 2005).

Since 2002, the OMG initiated a standardization process by issuing a Request for Proposal on Query / Views / Transformations “QVT”. This process will lead to an OMG standard for defining model transformations, which will be of interest not only for PIM-to-PSM transformations, but also for defining views on models and synchronization between models. Driven by practical needs and the OMG’s request, a large number of approaches to model transformation have recently been proposed. But they mainly are software oriented. In this paper, we propose to transform UML SD-to-PNs whatever the system type is, e.g. for heterogeneous systems. Some papers suggested the Petri net formalism for the modeling of the objects interaction (Paludetto et al. 2004). They proposed transformation rules at model level. We propose to bring the SD-to-PNs transformation at the meta-model level for only a part of the MOF. The reason why will be explained in section two here after.

This paper is made up with six sections. Thus, after this introductory first Section, Section two gives an overview of the context, foundation and limitation of the work. Some SD to PNs rules transformations are presented in Section three. Section four describes both source and target meta-models suited to the transformation. Section five proceeds to the design of the rules transformation meta-model. Finally, the approach retained is assessed and future prospects are outlined in the concluding Section.

2. CONTEXT AND FOUNDATION OF THE WORK

In the translation of sequence diagrams into PNs, the difficulty arises within the framework of the model transformations. Several efforts have addressed transformations at the

model level of UML with Petri nets (King and Pooley 2000), (Paludetto et al. 2004), but few have used meta-modeling. However the interest of the meta-modeling is well known now (Ferber and Gutknecht 1998), (Artikson 1997), (Trowitzsch et al. 2005). In this way it is possible to describe and classify the various concepts of a language or a model, and make easier the rules specification of the transformation. This also allows the mapping between the concepts of both meta-models source and target. For a given model, several meta-models may be defined. The suited one depends on the end purpose. Our objective focuses on the requirements V&V, a part based on formal properties and the rest by means of the simulation through VHDL-AMS language. All the necessary tools are layered in a platform according to the MDA approach (see Fig. 1).

This paper concerns the first transformation shown in the Fig. 1. As we said above it is based on a meta-modeling approach. The Fig. 2 models all the transformation process implemented in the platform. Only the first part (dimmed part of the Fig. 2) is discussed in this paper. As shown, the model level has been used only for checking off and to classify the transformation rules. Then, knowing the end purpose of the transformation, the incoming meta-models (SD and PNs) and the transformation meta-model itself may be built and implemented. The next sections will present the approach and the construction of these three meta-models.

When working with UML meta-model, people often refers to the OMG MOF because of its standard sight. This gives an ability to cover all language characteristics for various application domains. Due to its genericity meaning, the MOF is interesting for tool specifications but really heavy for modeling, specifically when the system dynamic must be described formally. In this work an amount of details given by the MOF are not all attractive for the objectives of such a modeling, especially for the requirements modeling. So, we refer to the MOF only for the interesting semantics of the SD at the high level modeling, saying message passing, method activation and data transfer. Indeed, the accordance with the whole MOF standard will be done later, after the validation of the approach.

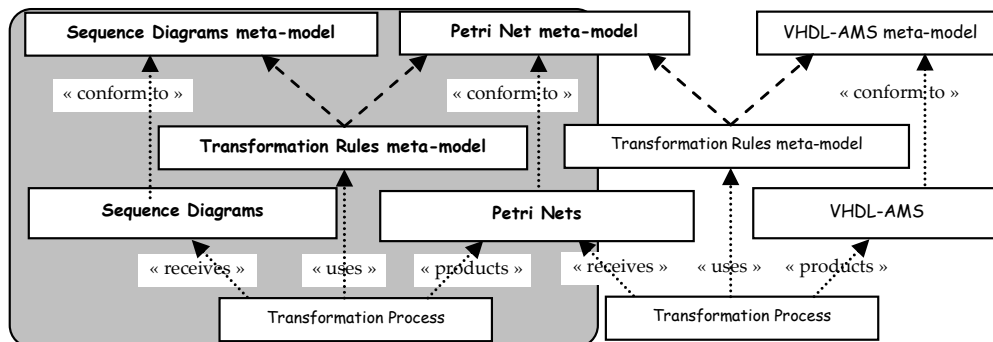


Fig.2. Meta-model based transformation architecture

3. SD INTO PN TRANSFORMATION TECHNIQUES: MODEL LEVEL

As previously mentioned, inter-object communication as well as method and abstract data give a high level view of the system requirements. So, in such a context, sequence diagrams into Petri nets transformation primarily depends on the type of the messages that link up the objects to each other. For instance, a message can be used to express a data transmission or a method call either with or without data transmission. In the same way, a message can wait or not either a result or an acknowledgement. This information can be known if the message semantics is written in accordance with the OCL language (Warmer and Kleppe 2003). This leads to translate synchronous or asynchronous communications into Petri nets.

Specifically we focus on inter-objects communications, as they are described at sub-system levels, and this section assumes either the following interpretations of an UML message: Asynchronous communication, Highly synchronous communication and Loosely synchronous communication.

3.1. Asynchronous communication

Fig. 3 shows a SD asynchronous message and its translation into an asynchronous communication. With Petri net model, such a communication is made up by means of a shared place that is seen as an outcome place from the sender object (*object1*) and an income place from the receiver object (*object2*); the sender (*object1*) and the receiver (*object2*) are represented each one as P-T-P (Place-Transition-Place) Petri net sequence. The different states of the message passing can be the following:

- From the sender point of view, a token on a first P of its P-T-P sequence means the beginning of the sending, the firing of the transition means the sending itself and a token in the second P, the end of the sending.
- From the receiver point of view, a token in the first P of its P-T-P sequence means that it is waiting for a message (token) on a shared place, the firing of the transition means the receiving itself and a token in the second P, the end of the receiving as well as the method running ("op").
- From the message itself, the beginning of the transaction takes place when the sender T is fired and the end when the receiver T is fired in turn.

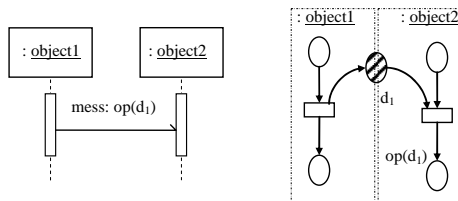


Fig. 3. Mapping of an asynchronous communication

From the system view, the sender puts a token in a shared place (*d1*) in order to require a method ("op", provided by the receiver), and the receiver accepts the call when its state

allows it to consume the *d1* token and, early following runs the called method "op".

3.2. Synchronous communication

Highly synchronous message is equivalent to the well known remote procedure call protocol (Fig. 4). The Petri net modeling of such a message is derived from the previous one where the PNs sequence of the sender as well the receiver is built with two merged P-T-P sequences in a P-T-P-T-P resulting sequence. Moreover, two shared places are implemented, one for the call and the second for the return. The centric P of the P-T-P-T-P sequence plays the part of waiting place for the sender and provided method place for the receiver. The second shared place is equivalent to the acknowledge return or result.

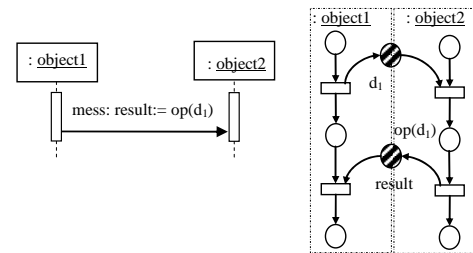


Fig. 4. Highly synchronous communication

The loosely synchronous message is like the highly one with a little bit difference. The sender does not need the result of the operation carried out by the receiver (Fig. 5); it just needs the acknowledgement of the request, and then it continues its own thread of control while the receiver performs its thread too that begin with the provided method ("op (*d1*)").

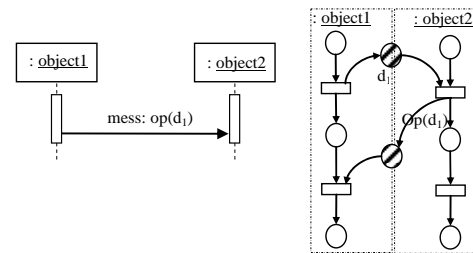


Fig. 5. Loosely synchronous communication

Note that the message syntax of the SD is the same for both asynchronous communication and loosely synchronous communication. In this case, the designer is responsible of the choice of the communication type that will be used in the design carrying on.

4. DEFINITION OF TARGET AND SOURCE META-MODELS

4.1. "Petri nets" meta-model

The objective of this work is to make easier the require-

ments validation process (see Fig. 1). For the best coherence of the whole transformation process, we used the Petri nets meta-model developed by (Albert et al. 2005) (Fig. 2). However, the working of this meta-model was limited to the representation of inter-objects communications. We extended the Petri nets meta-model in a way shown by Fig. 6.

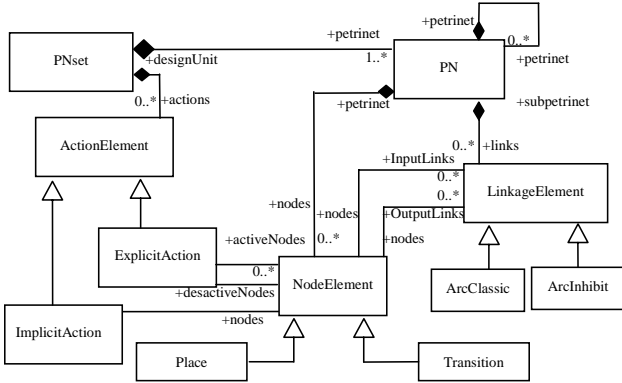


Fig. 6. PN meta-model for SD to PN transformation

As shown in Fig.6, *ArcClassic* and *ArcInhibit* classes both inherit from the *LinkageElement* class; only *ArcClassic* is considered in this paper. The same applies for *ExplicitAction* and *ImplicitAction* that inherits from the *ActionElement* superclass, and are associated to the *NodeElement*: only the second one will be considered in this paper (their activation modes are different: *ExplicitAction* is activated on positive/negative going edge of the associated node, *ImplicitAction* is simultaneously activated with the associated node).

4.2. “Sequence diagrams” meta-model

The objective of the sequence diagrams meta-model is the description of its model elements for their translation into Petri nets. The chosen structure (Fig. 7) describes the elements involved in object interactions.

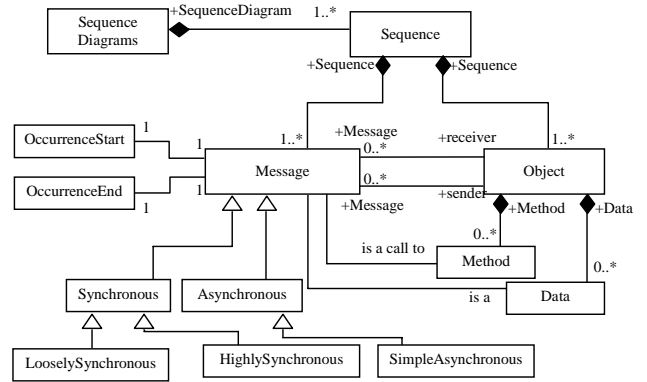


Fig.7. SD meta-model for the transformation of SD into PN

The inter-objects messages can be recognized in this structure as well as their type characterization, synchronous or asynchronous. They express a method call with data exchange. For later use, the edges triggering for start and end message (*OccurrenceStart* and *OccurrenceEnd* classes) are foreseen in order to take into account timed requirements of the communications. For instance, they will be useful for the modeling of the propagation delay of any communication. These timed requirements will not be considered in the transformation presented in this paper.

5. TRANSFORMATION RULES META-MODEL

5.1. The Expression of the Transformation Rules

The transformation rules meta-model (Fig. 8) describes the interactions that exist between classes of the “sequences diagram” meta-model (on the left part of the figure) and “Petri nets” meta-model (right part).

In this transformation, the information of method call with data transfer is associated with a shared place (or communication place *Pcom*) (see *dl* in Fig. 3), whereas the call of method is translated into *ExplicitAction* in relation to receiver description (see *op* (*dl*) in Fig. 3).

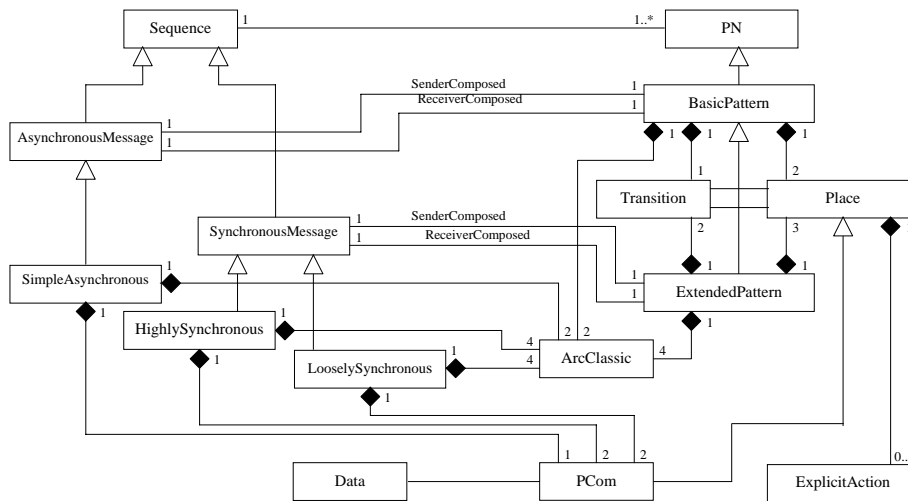


Fig.8. Transformation rules meta-model

In the same way, the result returned when method call is performed is a data associated to a shared place (see *result* in Fig. 4). To express these transformations we retained two Petri net patterns: a *basic* pattern, built with a sequence P-T-P (see *object1* of Petri net in Fig. 3) and an *extended* pattern for P-T-P-T-P sequence (see *objet1* in Fig. 4). These patterns are in relation with the type of message to be translated, either “AsynchronousMessage” or “SynchronousMessage”.

The meta-model distinguishes *Place* and *PCom* classes in order to clarify the transformation rules. Indeed, the places set of the Petri net that results of the message transformation contains the shared places. These later have a specific role: they play the part of an interface between the two communicating objects.

The classes of the transformation rules meta-model are characterized by the provided methods and attributes.

5.2. Execution of the transformation rules

The transformation rules are represented by the meta-model shown in section 5.1; it is important to explain the mechanism used for the transformation of the sequence diagrams into Petri nets.

Let us consider the transformation of an asynchronous message of the SD into its equivalent representation with Petri nets. The main steps of this transformation are: 1) create two single-patterns P-T-P for the two communicating objects (sender and receiver); 2) create the shared place and 3) connect the T sender to the shared place and the shared place to the T receiver. The transformation of the highly synchronous message and the loosely synchronous message into PN is done in similar way.

The transformation mechanism from a source model (SD) to a target model (PN) can be automated with an implementation process based on the Eclipse development platform (<http://www.eclipse.org/>). The benefit of such a platform is to provide an extensible, universal and versatile integrated development environment. It is possible to create development projects using a given language if this later is “connected” to the platform.

CONCLUSION

In this paper, we presented the meta-modeling based transformation of Sequences Diagrams into Petri nets in order to obtain a model available for the requirements validation. We saw that the meta-model approach opens the way towards a formal transformation and the result, saying PNs, allows the verification of system requirements, in our application. The approach is also interesting for to automate the transformation and to design tools in accordance.

However, many points remain to be deepened and need further investigation. For instance, time does not appear explicitly, but implicitly by the way of order relation. In future work we will consider the explicit time with an aim of taking into account the asynchronous communications with propagation delay. Then, it will be suitable to use the timed Petri net model and thus to enrich the proposed meta-models.

For now, we considered only one sequence diagram. One would think that it is reasonable for the highest level of ab-

straction, since this sequence diagram may represent the main scenario. But, when considering complex system it is necessary to transform several sequence diagrams using a successive refinement method within an incremental approach: so it is essential to consider the behavior and the interactions of implied objects in several use cases. Such an approach will imply PNs composition.

For the systems of our interest, the use cases generally correspond to operating modes: an object has often several operating modes. The description of the global behavior of the object will be carried out by modes composition (scenarios), after the separate validation of the different modes concerned by this object. The validation of all the modes for a given object will be done by the validation of the resulting Petri net. This task is made easier by the simplicity of the Petri nets patterns implied in the composition (well-structured blocks).

REFERENCES

- Albert, V., N'ketsa, A. and Pascal, JC.: “Towards a meta-model based approach for hierarchical Petri net transformations to VHDL”. *2005 European Simulation and Modeling Conference*, Porto (Portugal), October 24-26, 2005, pp.531-536
- Artikson, C.: “Meta-Modeling for distributed Object Environments”. *IEEE Enterprise Distributed Object Computing Workshop*, Oct. 24-26, 1997, pages 90-101, Gold Coast, QLD.
- Czarnecki, K. and Helsen, S.: “Classification of Model Transformation Approaches”, *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*
- Esteban, P., Ouadani, A., Paludetto, M. and Pascal, JC.: “A component based approach for system design and virtual prototyping”. *12th Annual European Concurrent Engineering Conference (ECEC'2005)*, Toulouse (France), April 11-13, 2005, pp.85-90.
- Ferber, J. and Gutknecht, O.: “A meta-model for the analysis and design of organizations in multi-agent systems”. *IEEE Multi Agent Systems*, July 3-7, 1998, pages. 128-135, Paris.
- Frankel, D.: *Model Driven Architecture: Applying MDA to Enterprise*. Computing Wiley Press, 2003
- Gerber, A., Lawley, M., Raymond, K., Steel, J. and Wood, A.: “Graph Transformation”, *First International Conference (ICGT 2002)*, Barcelona, Spain, October 7-12, 2002. Proceedings. LNCS vol. 2505, Springer-Verlag, 2002, pp. 90 - 105
- King, P. and Pooley, R.: “Derivation of Petri Net Performance Models from UML Specifications of Communications Software. Computer Performance Evaluation, Modeling Techniques and Tools”, *11th International Conference, TOOLS 2000*, Schaumburg, IL, USA, March 2000. Proceedings / Boudewijn R. Haverkort, Henrik C. Bohnenkamp, Connie U. Smith (Eds.) -Springer Verlag, 2000, pp. 262-276.
- Kleppe, A., Warmer, J. and Bast, W.: *MDA explained: The Model Driven Architecture, Practice and Promise*. Addison Wesley, 2003
- Paludetto, M., Delatour, J. and Benzina, A.: “UML et réseaux de Petri - Vers une formalisation des besoins des systèmes embarqués”. *Technique et science informatiques "TSI"*, page n° 543 Fascicule N° 4, Vol N°23, 2004.
- Trowitzsch, J., Zimmermann, A. and Homme, G.: “Towards Quantitative Analysis of Real-Time UML Using Stochastic Petri Nets”. *19th Parallel and distributed processing symposium IPDPS'05 “IEEE”*, April 04-08, 2005.
- Warmer, J. and Kleppe, A.: *The Object Constraint Language Second Edition: Getting your Models for MDA*. Addison Wesley, 2003.