Using Petri Nets to Specify Collaborative Three Dimensional Interaction

Rafael Rieder[1], Márcio S. Pinho[1], Alberto B. Raposo[2]
[1] *Pontifical Catholic University of Rio Grande do Sul, PUCRS, Faculty of Informatics*
[2] *Pontifical Catholic University of Rio de Janeiro, PUC-Rio, Department of Informatics*
{rafael.rieder, marcio.pinho}@pucrs.br, abraposo@inf.puc-rio.br

Abstract

This work presents a methodology to formally model and to build collaborative three dimensional interaction tasks in virtual environments using three different tools: Petri Nets, Interaction Technique Decomposition taxonomy and Object-Oriented concepts. The user operations in the Virtual Environment are represented as Petri Net nodes and these nodes, when linked, represent the interaction process stages. The integration of these approaches results in a modular application, based on the Petri Nets formalism that allows for specification of collaborative interaction tasks, and also the reuse of developed blocks in new virtual environment projects.

Keywords: collaborative interaction, interaction technique specification, design process.

1. Introduction

Research on collaborative manipulation of objects in immersive virtual environments (VEs) is relevant in many areas, such as simulation and training, as well as in data exploration [1]. Collaborative manipulation, or collaborative three-dimensional interaction, refers to the simultaneous manipulation of a virtual object by multiple users in a VE. In simulation and training, simultaneous manipulation of objects in VEs can be used to mimic some aspects of real-world tasks. For example, in situations like product and equipment design, assembly tasks or emergency training, even when the users are not co-located in space, collaborative manipulation may provide more realistic interaction. In data exploration, collaborative manipulation is an important tool to enhance the interaction process, by moving it from being one-sided ("I do this, while you watch") to being truly collaborative, increasing insight exchange and reducing the time for task completion.

In order to better understand a virtual reality (VR) application, especially its possible intricate interaction flow, it is very helpful to use some kind of formal description tool like Petri Nets (PN) that can describe the system function and components. This allows not only a better understanding, but also a preliminary evaluation of each phase of the system operation, which is especially useful in collaborative 3D interaction, since the collaborative metaphor concept needs the representation of parallel activities. Moreover, a formal description facilitates the automatic generation of the core application code, from graphical representations.

Besides formal specification tools, some researchers have sought to develop taxonomies able to document and specify VEs in an abstraction level closer to the user's conception instead of the programmer's view of the application [2, 3]. These approaches split the systems into smaller parts, identifying behavior patterns and allowing to encapsulate them into classes that are able to execute some relevant functionality. This approach allows for the reuse of these classes in other projects and also allows for the combination of them to build a new interaction technique, for example.

Both the use of formalisms and taxonomies aim to better define the interaction processes, reducing the time spent during the design and implementation of VEs. Therefore, an integration of both approaches can gather the best of them: system specification according to the user's level of expertise, evaluation in early stages of the development process, and the detailing of each phase of the software development process.

This paper describes a methodology able to model and to implement software modules that represent the collaborative interaction process phases. Our methodology integrates three modeling approaches: PN formalism [4], a collaborative manipulation model [5] based on the combination of Bowman's single user interaction techniques taxonomy [2], and object oriented programming concepts. The combination of these elements allows for the description of interaction tasks, in which the sequence of the interaction processes is controlled by PNs, and whose codes are generated automatically. By the integration of a collaborative interaction techniques taxonomy, the formalism of PN and automatic code generation, the present work addresses the entire development cycle of a collaborative three dimensional interaction.

This paper is organized as follows. Section 2 summarizes work that is related to our approach. In Section 3 we present the proposed methodology and in Section 4 we present a case study with a collaborative manipulation task. Section 5 concludes the paper.

978-1-4244-3535-7/09/$25.00 ©2009 IEEE

2. Background

This section presents related work in four aspects that are related to our approach: interaction techniques specification; interaction techniques taxonomies; collaborative manipulation; and frameworks and tools for code generation in VR applications.

2.1. Interaction Technique Specification

Smith and Duke [6] point out that the lack of formal descriptions during the development process of VEs inhibits the identification of similarities among different interaction techniques, leading to the "reinvention" of existing techniques. Interaction technique specification is an important task from the perspective of both users and designers. Users require interaction techniques that allow them to complete interaction tasks in a particular application and designers like to build systems that make the required interaction possible [7].

Hynet [8] is a specification methodology for interaction techniques that integrates three modeling approaches. High-level PNs represent the formal base for the specification, defining the application semantics and allowing a graphical representation for the application events (the discrete part of the application). Differential Algebraic Equations handle the continuous behavior pattern of the application, and Object Oriented Concepts allow for the enhancing of the methodology expressiveness, generating concise and compact models.

Based upon HyNet, the Flownet methodology [9] was developed for describing dynamic behavior patterns in VEs and presents a different graphical notation that allows for the specification of both the discrete and continuous behavior patterns of the application.

The Interactive Cooperative Objects (ICO) is a formal notation devoted to the specification of interactive systems [10]. It borrows concepts from the object-oriented programming to describe the structural or static aspects of systems, and uses high-level Petri nets to describe their dynamic aspects. According to the authors, the specification created using ICO can be simulated, which gives the possibility to prototype and test an application before it is fully implemented.

The above specification approach provides systematic methods for interaction techniques design, test, and refining, facilitating the description of systems. However, both HyNet/ Flownet and ICO are not related to any interaction technique taxonomy and do not provide any support for the automatic generation of code, which requires a deeper knowledge of the used formalisms, both by the designers and the developers, especially in the implementation phase.

2.2. Interaction Technique Taxonomies

We can view the development process of VEs under the perspective of the interaction techniques used, with the aim of classifying them in order to better understand their components, and therefore the possibilities of software reuse in new applications.

Lindeman [11], for example, demonstrates a taxonomy that divides interaction techniques according to the type of manipulation technique (direct or indirect), the system actions (discrete or continuous) and the degrees of freedom controlled by the interaction technique. This approach helps to identify the parameters involved in each interaction technique, facilitating the building of new forms of interaction.

Bowman *et al*. [3] present a taxonomy based on task decomposition to perform a detailed analysis of the interaction process. According to them, the separation of tasks in simpler modules allows each of them to be analyzed and tested in an independent way as a tool for evaluating the usability and effectiveness of an interaction technique in a particular context or VE.

Another advantage in the use of a task decomposition taxonomy is the possibility of reuse or combination of interaction technique components in new projects. This characteristic allows, besides flexibility, for the conception of new techniques adequate for specific situations, such as collaboration.

2.3. Collaborative 3D Interaction

In most of the known collaborative virtual environments, the simultaneous manipulation of the same object by multiple users is avoided. True collaborative manipulation has been the focus of a few research efforts.

Earlier research by Ruddle *et al*. [1] presented the concept of rules of interaction to support symmetric and asymmetric manipulation. In a subsequent work [12], the same authors separated collaborative tasks into two levels of control. The high level control activities correspond to those tasks that require attention, planning and mental effort by the users to be executed. The low level control activities are quasi-autonomous activities that, once learned, are quickly executed by the users with no conscious control.

The work of Pinho *et al*. [5] presents the concept of collaborative metaphor for simultaneous interaction in VEs. This metaphor is composed by a set of rules that defines how to combine each step of the interaction process, allowing that interaction techniques normally used in individual interaction be combined to compose a collaborative technique. The steps of the interaction process used are those steps defined by Bowman's task decomposition taxonomy [3]. The combination of the steps of interaction techniques is obtained by the distribution of the degrees of freedom of the objects' control among the users. We follow this methodology to describe collaborative 3D interaction in this work.

2.4. Tools for Code Generation

VR Frameworks try to separate functions in modules, allowing for the abstraction of the complexities of some system actions, and furthermore, the reuse of these software modules.

The Unit framework [13] inserts an abstraction layer between the applications and its devices, and inserts application units into a data flow. The code generation process offered by Unit, however, results in interpreted code, which may compromise the quality of the interaction if the hardware does not support the application demands.

Using a different perspective, Ying and Gračanin [14] aim to understand the interaction process of existing VR applications. Analyzing an existing code, the information related to the user interaction is extracted and organized in an XML file that serves as a base for building a PN model representing the target application. By simulating the PN, one can "view" the interaction process through the PN behavior pattern, while the user is interacting with the VR application. Nevertheless, this approach is restricted to the test phase and does not contemplate previous steps of the software development process, because reverse code engineering is used to create description files.

3. The Proposed Methodology

From the literature review, it is possible to conclude that the approaches presented above have specific advantages and goals. However, in general, they do not address the entire computer application development cycle, especially concerning the final phases of debugging and code generation. Towards this goal, we developed a methodology for hierarchical development of a VR collaborative interaction process using Petri Nets, beginning from the design stage, based upon Bowman's interaction taxonomy, up to the implementation phase, relying on the object oriented programming paradigm.

The main goal of the proposed methodology is to model and implement modules that represent the steps of the interaction process. The use of formalism in conjunction with an interaction taxonomy allows for the detailed specification of the system, as well as facilitating the structuring and implementation process, encapsulating functionalities. These characteristics enable the generated modules to be tested in advance and reused later, simplifying and accelerating the development of VEs.

The graphical representation adopted here is based on Colored Petri Nets (CPNs) [15] because, during the modeling process, we need an easy way to differentiate the various types of data that are manipulated in a VR application. For simplification, this work refers to CPNs simply by the expression Petri Nets (PN).

The methodology presented in this paper aims to approximate the user's conception of the application from the designer and developer's point of view, modeling the application under the perspective of tasks which the user has or wishes to perform inside the VE. These tasks can be decomposed in elementary-tasks that can be easily identified in most VR applications [2]. These elementary-tasks split the interaction process into three phases: *selection*, *manipulation* and *release*. Our work adapts this taxonomy dividing the selection phase into *selection* and *attachment*. The former represents the indication of the object which the user wishes to manipulate. The latter deals with the confirmation of this selection. Both provide feedback to the user in order to confirm their execution.

The designer needs to follow three steps to apply the methodology: 1) identify the VE tasks, according to Bowman's taxonomy, as well as the main states reached by the application after executing each task; 2) define a PN with the tasks and states identified in the previous step; 3) Implement the model, using a set of classes specially developed to build the PN and to control its execution. Each of the above phases is detailed below.

In order to illustrate the use of our methodology in the specification process of a VE, we built a virtual living room application in which the user's primary goal is to organize the furniture. In this section we present a single user interaction, and in the following section we extend it to collaborative interaction.

3.1. Identifying Interaction and Building the PN Model

The first phase of our methodology identifies the application phases based on Bowman's taxonomy. The application starts in the *Selection State* (Figure 1, on the left) in which the user can move the pointer, looking for an object to select. From this point the *Selection Task* tests if there is a collision between the pointer and a object. If so, the *Selection Task* transition is fired, and the *Attachment State* is established.

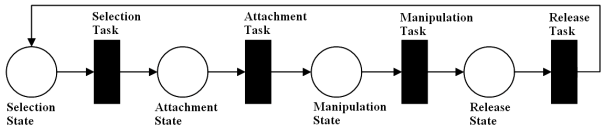

Figure 1. A high-level PN for the application.

At this point if the user presses and holds the selection button, the *Attachment Task* is fired attaching the selected object to the pointer and establishing the *Manipulation State*. Once this state is established the PN fires the *Manipulation Task* which allows the user to relocate the object using the pointer. If the user releases the selection button, the *Release State* place enables the firing of *Release Task*, separating the pointer from the previous selected object.

After identifying the application tasks in a high abstraction level, it is necessary to perform a task

subdivision process, splitting them into smaller parts (see Table 1), based upon the operations each of them has to execute.

Table 1. High-level tasks detailing.

High-Level Tasks	Basic Operations
Selection	– Indication Subtask – Indication Feedback Subtask
Attachment	– Confirmation Subtask – Confirmation Feedback Subtask
Manipulation	– Positioning Subtask
Release	– Detachment Subtask – Detachment Feedback Subtask

Following the methodology, the identification of the necessary resources (data) for each interaction phase should be initiated, which is represented as tokens in the PN. For this, PN arcs should be labeled with the token types, graphically represented by icons.

The places *Selection State*, *Attachment State*, *Manipulation State*, and *Release State* need to be constantly updated with information about the devices and control variables from the application. Therefore, tokens with these data must be inserted into them, as can be seen in Figure 2 that presents the complete PN model for the application. Utilizing formalism, interaction devices and the application can be represented as source transitions in the PN model, as they don't have input places, being always enabled to fire and to produce tokens to the net (in this case, information about the user physical interaction and the VE state). In Figure 2 the devices are represented by triangles, while the application is represented by hexagons. These shapes are merely illustrative and serve only to help the understanding of the network.

The pace of the PN simulation is controlled by the application, which tests each transition every time the user's view needs to be modified, guiding the execution of actions. In this step, the simulator fires the transitions that have their pre-conditions fulfilled. A transition's firing generates a call to a function previously assigned to the task. In other words, all the transitions are checked on every rendering cycle and fired or not, depending on the existence of the pre conditions (the necessary tokens). Therefore, it is possible to analyze the process logic together with the system and devices behaviour patterns.

3.2. Implementation Phase

In order to derive the implementation, we start with the graphical description of the PN, created in Dia editor (http://live.gnome.org/Dia). From this diagram, an XML specification is obtained which, in turn, originates a C++ code.

Using Dia, it is possible to add support for new types of diagrams by writing simple XML files, thereby creating specific libraries with elementary objects called "shapes". This feature also allows diagrams to be stored in XML files, facilitating the conversion of models to other codification forms, such as Java and C++ languages, or other markup languages, such as PNML (Petri Net Markup Language) [16]. The flexibility of the conversions allows for designers the reuse of models in new projects or different development platforms, and the use of other tools able to validate the syntactical structure of PNs.

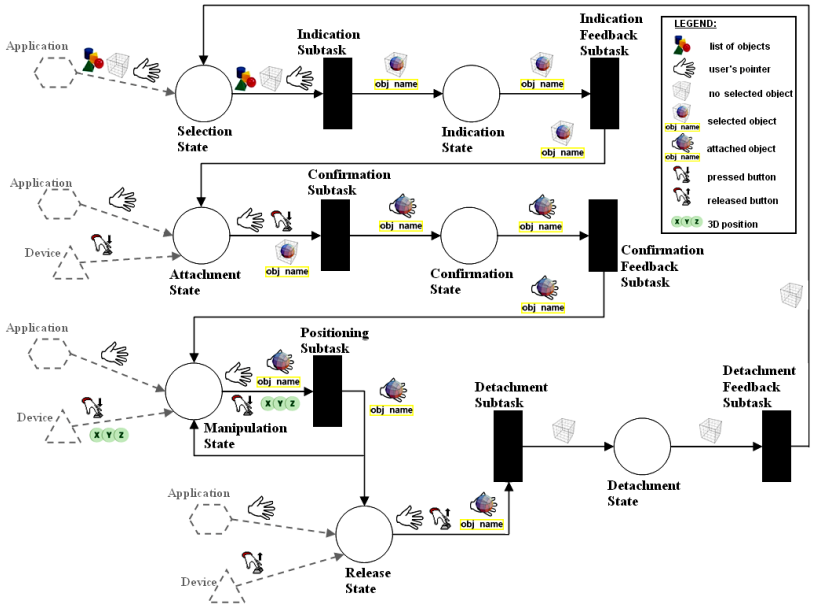

Figure 2. PN model and data resources for the modelled interaction process.

Dia uses a XSLT (eXtensible Stylesheet Language Transformation) file format that allows conversion of XML tags to construction of a programming language, such as C++. We created two XSLT files that facilitate this conversion. The first file defines rules to generate an XML file from the graphical diagram, whereas the second contains rules to convert from the PN elements to C++ classes. These classes run the modeled PN and may be connected to a VE, coordinating its interaction.

4. Collaborative Manipulation Case Study

We adapted the model of the virtual living room application in order to support collaborative manipulation tasks. Figure 3 presents the new model, including two new tokens responsible for the objects' translation and rotation tasks. This way, for instance, during the manipulation step an user may control the positioning of an object in the X-Y plane, while another user may control its orientation in the Y axis.

Each user interacting in the application has his own tokens, since the PN represents the behavior of the system as a whole. In Figure 4, for example, tokens representing each user have a specific border color (blue and red), while the shared token, representing the object has a different border color (magenta). In this example, the blue user is responsible for the object's translation in the 3D space, while the red user is responsible for the rotation of the same object. For this example, we considered the arcs label definition presented in Figure 3.

It is possible to show that the net also provides support for the representation of individual interaction. Figure 5 presents interactions of two users with different objects, in distinct steps of the interaction process (selection and attachment).

In comparison with the standard approach to deal with collaborative 3D manipulation, our methodology approximates the application modeling to the user's conception, allowing effective communication between designers, developers and end-users during the entire development cycle. The PN interaction process representation also facilitates the identification of the parts of the system that could be parallel or support similar situations, such as individual and collaborative interaction. This can result, for example, in simplification of the system and reduction of the design-time and development-time, since our methodology is based on the top-down development approach, which allows breaking down a system to gain insight into compositional sub-systems. So, it is possible to create interaction technique components from generic parts, and reuse them in future projects.

However, our methodology still requires designers to know the basic PN concepts and rules to draw and to revise their models, since Dia has no native supported to PN projects. Depending on the system complexity, a training step could be needed to prepare the professionals to the PN model design and review.

Our intention is developed a mechanism able to verify the PN model correctness, allowing the PN validation before the automatic code generation. As a result, the PN project could be consistency and completeness, mainly to complex VE models.

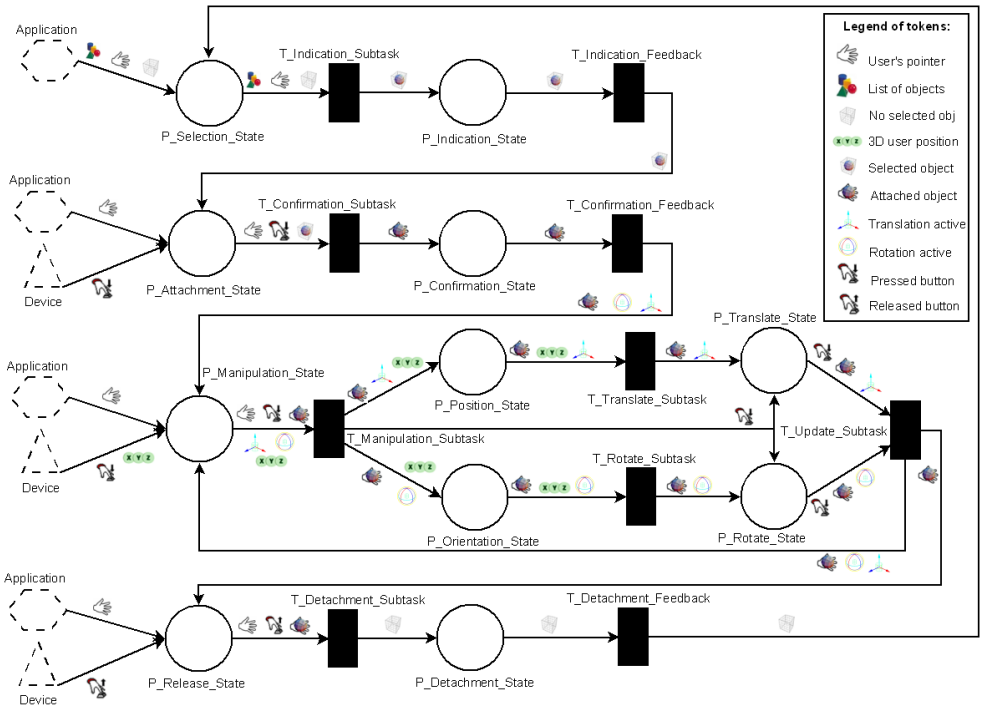

Figure 3. PN with parallel activities. "Rotation active" and "Translation active" tokens define the task to be executed.

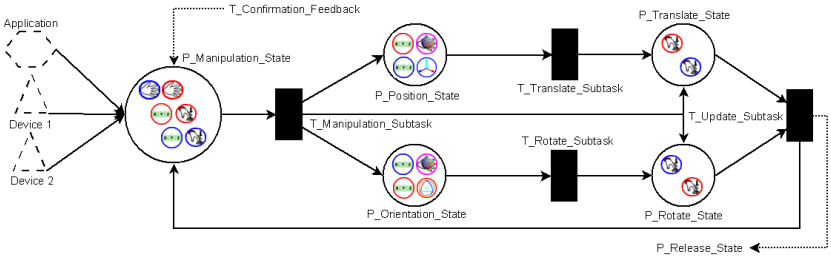

Figure 4. PN state during the cooperative manipulation.

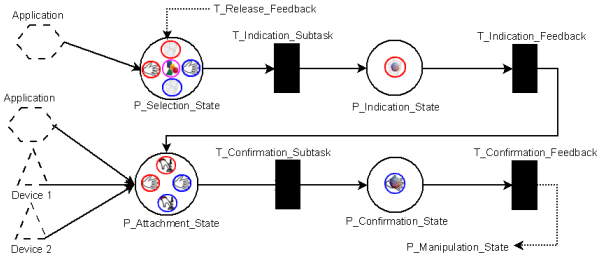

Figure 5. PN state representing the interaction with two distinct objects, in different tasks.

5. Conclusion

This work presents a methodology to specify collaborative interaction tasks for VR applications using the Petri Nets formalism as a base for the software design. The methodology aims to facilitate the application development from conception and design phases to the implementation, test and documentation processes. The option of code generation from the graphical model helps the communication between designers and developers, avoiding the breaking of paradigms and speeding up the development process. Moreover, an application built with our methodology can offer optimized functions, allowing users to complete their interaction tasks in an intuitive way.

Acknowledgements

This work was partially funded by Tecgraf, Computer Graphics Technology Group, at PUC-Rio. We are also grateful for the fellowships granted by Dell/PUCRS Agreement and CAPES - the Brazilian Ministry of Education Agency.

References

[1] R. A. Ruddle, J. C. Savage, D. M. Jones. "Symmetric and Asymmetric Action Integration During Cooperative Object Manipulation in Virtual Environments". *ACM Transactions on Computer-Human Interaction*, 9(4), 2002, pp.285-308.

[2] D. A. Bowman, L. F. Hodges. "Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments". *Journal of Visual Languages and Computing*, 10(1), 1999, pp. 37–53.

[3] D. A. Bowman, E. Kruijff, *et al*. *3D User Interfaces: theory and practice*. Addison Wesley, 2005.

[4] T. Murata. "Petri nets: Properties, analysis and applications". *Proc. of the IEEE*, 77(4), 1989, pp. 541-580.

[5] M. S. Pinho, D. A. Bowman, C. M. S. Freitas. "Cooperative object manipulation in collaborative virtual environments". *Journal of the Brazilian Computer Society*, 14(2), 2008, pp. 53-67.

[6] S. Smith, D. Duke. "Virtual Environments as Hybrid Systems". Proc. Eurographics UK 17th Annual Conference (EG-UK'99), 1999, pp. 113-128.

[7] S. Smith, D. Duke. "Using CSP to Specify Interaction in Virtual Environments". *Technical Report YCS 321, University of York – Department of Computer Science*, 1999.

[8] R. Wieting. "Hybrid High-Level Nets".Proc. 28th conference on Winter Simulation (WSC), 1996, pp. 848-855.

[9] S. Willans, M. D. Harrison. "Prototyping pre-implementation designs of virtual environment behaviour". Proc. 8th IFIP Int. Conf. Engineering for Human-Computer Interaction (EHCI), Springer-Verlag, 2001, pp. 91-108.

[10] P. A. Palanque, R. Bastide. "Synergistic Modeling of Tasks, Users and Systems Using Formal Specification Techniques". *Interacting with Computers*, 9(2), 1997, pp. 129-153.

[11] R. W. Lindeman. "Bimanual Interaction, Passive-Haptic Feedback, 3D Widget Representation, and Simulated Surface Constraints for Interaction in Immersive Virtual Environments". *PhD thesis*, F. School of Engineering and Applied Science, George Washington University, 1999.

[12] R. A. Ruddle, J. C. Savage, D. M. Jones. "Levels of control during a collaborative carrying task". *Presence: Teleoperators and Virtual Environments*, 12(2), 2003, pp. 140-155, 2003.

[13] A. Olwal, S. Feiner. "Unit: modular development of distributed interaction techniques for highly interactive user interfaces". Proc. 2nd GRAPHITE, 2004, pp. 131-138.

[14] J. Ying, D. Gračanin. "Petri Net Model for Subjective Views in Collaborative Virtual Environments". Proc. 4th Int. Symp. on Smart Graphics, LNCS 3031, Springer-Verlag, 2001, pp. 128-134.

[15] K Jensen. *Coloured Petri Nets: basic concepts, analysis methods and practical use.* Springer, 1997.

[16] J. Billington, S. Christensen, *et al*. "The Petri Net Markup Language: concepts, technology, and tools". Proc. 24th Int. Conf. of the Applications and Theory of Petri Nets (ICATPN), LNCS, Springer-Verlag, 2003, pp. 483–505.