Standardisation des réseaux de Petri : état de l'art et enjeux futurs

Lom M. Hillah[1] et Laure Petrucci[2]

[1] LIP6, CNRS UMR 7606 et Université Paris Ouest Nanterre La Défense
Lom-Messan.Hillah@lip6.fr
[2] LIPN, CNRS UMR 7030, Université Paris XIII
Laure.Petrucci@lipn.univ-paris13.fr

Résumé.

La communauté travaillant sur les réseaux de Petri a constaté relativement tôt le manque d'homogénéité dans les définitions formelles et les outils. En effet, il existe une grande variété de types de réseaux de Petri, avec des différences parfois significatives (par exemple les réseaux temporels) ou de simples extensions (les réseaux avec arcs inhibiteurs). De plus, chaque outil utilise sa propre version de réseau de Petri et implémente des techniques d'analyse qui lui sont propres.

La vérification de systèmes modélisés avec des réseaux de Petri nécessite l'utilisation de diverses techniques de vérification, donc des divers outils associés. Pour que cette démarche soit réalisable dans la pratique, la conception d'un format d'échange de réseaux de Petri entre ces outils est un enjeu majeur.

La norme ISO/IEC-15909, que nous présentons dans cet article, a pour objectif de répondre à cette problématique. Elle est structurée en 3 parties.

La première partie concerne les définitions du formalisme, donnant ainsi une sémantique formelle à plusieurs types usuels de réseaux de Petri : les réseaux places/transitions et les réseaux de haut niveau (colorés). Cette partie a été publiée comme norme internationale en décembre 2004. Un amendement définissant les réseaux symétriques devrait être bientôt publié.

La seconde partie, parue en novembre 2009, se concentre sur la syntaxe et l'élaboration d'un langage d'échange de réseaux de Petri. Ce langage, Petri Net Markup Language (PNML), s'appuie sur les concepts introduits dans la première partie du standard. Sa conception repose sur des métamodèles en Unified Modeling Language (UML), puis une transformation en eXtensible Markup Language (XML).

Un des enjeux majeurs de la seconde partie a été la flexibilité et l'extensibilité du langage PNML, en vue des travaux sur la troisième partie de la norme. Le développement de cette troisième partie vient de démarrer. Elle porte sur les extensions de réseaux de Petri. Ces extensions concernent la structuration des modèles (modularité et hiérarchie) et les mécanismes nécessaires à l'intégration de nouveaux types de réseaux de Petri et l'ajout de nouveaux éléments (tels que les arcs inhibiteurs).

Mots-clés : PNML, normalisation réseaux de Petri, métamodélisation, UML, MDE.

1 Introduction

Les réseaux de Petri constituent une famille de formalismes mathématiquement fondés, adaptés à la modélisation de systèmes parallèles et asynchrones [1]. Ils permettent l'analyse de différents aspects comportementaux d'un système (discret, stochastique, temporisé), soit sur des questions relatives à l'expressivité, soit sous l'angle de la décidabilité pour résoudre des problèmes de vérification [2]. Il en existe de ce fait plusieurs types, dont les sémantiques d'exécution diffèrent parfois de manière significative (p. ex., temporels vs. places/transitions ordinaires), ou représentent de simples extensions d'autres types (places/transitions avec arcs inhibiteurs).

L'écosystème des réseaux de Petri pourrait donc être caractérisé par les facettes suivantes :

- les grandes familles à sémantique discrète, à temps continu ou stochastique,
- les types au sein de chaque famille, ainsi que leurs extensions,
- la modularité et la hiérarchie qui peuvent être transversales à ces types,
- les divers formats de représentation des modèles de chaque type, propres aux outils.

Les formats font partie de langages de représentation propriétaires pouvant définir des « sucres syntaxiques » sur des modèles théoriques, aboutissant ainsi à la création de sous-classes hybrides. Leur interprétation, parfois sélective en fonction de l'objectif de vérification, repose sur l'adaptation singulière de chaque outil ou sur des convertisseurs dédiés au sein de plates-formes spécialisées. Cette souplesse technique, associée à la richesse sémantique du formalisme, rend finalement difficile la construction d'une démarche de vérification basée sur un

large spectre de modèles de types compatibles, de méthodes et d'outils.

L'hétérogénéité qui en résulte entre les types et les outils d'analyse sachant les manipuler donne souvent lieu à une incompatibilité sémantique. Conduire une démarche d'analyse nécessitant plusieurs techniques de vérification (et donc des outils correspondants) dans un tel contexte représente un véritable défi. Chaque outil manipulant son propre format associé à sa propre version de réseau de Petri, l'échange standardisé et automatisé de spécifications entre les outils devient de ce point de vue primordial. La norme ISO/IEC-15909 a pour objectif de fournir le cadre formel et unifié de définitions et techniques de représentation pour répondre à cette problématique.

ISO/IEC-15909 est structurée en trois parties. La première partie fournit le cadre sémantique formel unifiant les définitions des réseaux places/transition et des réseaux de haut niveau (à la Jensen [3]). C'est une norme officiellement publiée depuis décembre 2004. Ce cadre sémantique pose les bases pour l'élaboration d'un langage d'échange, créé dans la deuxième partie, officiellement parue en novembre 2009. Ce langage, Petri Net Markup Language (PNML), a été conçu selon les techniques de métamodélisation de *Domain-Specific Languages* couramment mises en œuvre dans *Model Driven Architecture* (déclinaison industrielle par l'Object Management Group du paradigme de développement dirigé par les modèles - *Model-Driven Development*).

La troisième partie dont les travaux sont en cours, doit mettre en place un cadre pour les extensions des réseaux de Petri. Ces extensions portent d'une part sur leur structuration modulaire et hiérarchique et d'autre part sur les techniques nécessaires à la définition et l'intégration de nouveaux types, ou de simples extensions de types existants. Les nouvelles définitions des utilisateurs de la norme, dès lors qu'elles respecteront les mécanismes mis en place par la troisième partie, feront alors partie intégrante de la norme.

Nous présentons dans cet article l'état de l'art sur les travaux ayant été accomplis et actuellement en cours autour la norme. La section 2 sera l'objet de cet état de l'art et du retour sur expérience que nous en avons tiré. Ce retour d'expérience pourrait être utile à la communauté dans la détermination des prochains grands chantiers théoriques, techniques et technologiques autour des réseaux de Petri et dans la manière de les mettre en œuvre. La section 3 présentera l'implémentation de la norme par un outil suivant les techniques de *Model-Driven Engineering* et l'effort de dissémination entrepris par le groupe de travail sur la norme au sein de la communauté. Cet outil libre et gratuit, PNML Framework, entend favoriser l'adoption et l'intégration rapide de la norme au sein des outils d'analyse. La section 4 dresse le tableau des enjeux futurs et de l'approche adoptée en vue de leur résolution. Enfin, la section 5 conclut cet article.

2 État de l'art de la norme et retour sur expérience

L'idée principale de la norme est de capturer et d'expliciter formellement la relation sémantique et syntaxique qui existe entre les différents types de réseaux de Petri. Le formalisme ayant grandi par des définitions particulières et leurs extensions au cours du temps, il est nécessaire après un demi-siècle de maturité théorique d'unifier l'existant pour faciliter les échanges. La première partie de la norme s'est donc intéressée à la famille des réseaux de haut niveau, car elle était sans doute la plus répandue et sa définition bénéficiait d'un fort consensus au sein de la communauté.

2.1 Première partie : modèle sémantique des réseaux de haut niveau

Le modèle sémantique et la notation graphique des réseaux de haut niveau (parmi lesquels les réseaux colorés) et des réseaux places/transitions (où les jetons appartiennent à une seule classe de couleur, *black dot*) ont été définis dans un premier temps. Les réseaux symétriques (ou réseaux colorés bien formés) en cours de formalisation dans un amendement à la première partie, sont définis comme une sous-classe des réseaux de haut niveau. Ils n'utilisent qu'un sous-ensemble des types de données algébriques (types finis, ainsi que les fonctions associées) des réseaux de haut niveau. Les types de données utilisateurs sont créés sur cette base, mais les fonctions arbitraires ne sont pas autorisées. Grâce à sa capacité de représentation de symétries comportementales dans les modèles, ce type de réseaux de Petri permet la construction du graphe d'accessibilité symbolique, efficace pour combattre le phénomène récurrent d'explosion combinatoire pendant la vérification.

Le modèle sémantique des réseaux de Petri de haut niveau est brièvement défini de la façon suivante :

Un réseau de Petri de haut niveau est le tuple *HLPN =(P, T, D, Type, Pre, Post, M_0)*, où :

- *P* est un ensemble fini d'éléments appelés Places
- *T* est un ensemble fini d'éléments appelés Transitions, disjoint de P (i.e., $P \cap T = \emptyset$)
- *D* est un ensemble fini non vide de domaines non vides, où chaque élément de D est appelé type
- *Type :* $P \cup T \rightarrow D$ est une fonction qui, à toute place et toute transition, associe un type. Pour les transitions,

le type définit l'ensemble des modes de la transition.

- *Pre, Post : TRANS* → *µPlace* sont les pre et post mapping, avec :

o *TRANS = {(t,m) | t ∈ T, m ∈ Type(t)}*

o *PLACE = {(p,g) | p ∈ P, ∈ Type(p)}*

- $M_0 \in \mu PLACE$ est un multi-ensemble appelé marquage initial du réseau. *µPLACE* est décrit comme l'ensemble des multi-ensembles de *PLACE*.

Un graphe de réseau de Petri de haut niveau est constitué des éléments suivants :

- Un graphe biparti, comprenant deux ensembles de nœuds, les places et les transitions, ainsi que les arcs reliant les places aux transitions et vice versa.
- Le type (ou domaine) des places, qui est un ensemble non vide. Un type est associé à chaque place.
- Le marquage des places, qui est une collection d'éléments appartenant au type de la place, associé à la place. Dans cette collection, la répétition des éléments est autorisée. Les éléments sont appelés jetons.
- L'annotation des arcs, qui est une expression associée à l'arc, comprenant des constantes, des variables et des images de fonction (e.g., f(x)). Les variables sont typées. Les expressions sont évaluées en assignant des valeurs aux variables. L'évaluation de l'expression associée à un arc résulte en une collection d'éléments appartenant au type de la place reliée à l'arc. Cette collection est un multi-ensemble.
- La condition des transitions, qui est une expression booléenne associée à chaque transition.
- Les déclarations du réseau, qui comprennent les définitions des domaines des places, des variables ainsi que les définitions des fonctions.

La première partie définit des niveaux de conformité, permettant aux utilisateurs de situer leur niveau d'implémentation de la norme. Il existe quatre niveaux de conformité, regroupés en deux ensembles de deux niveaux. Le premier et le deuxième niveaux concernent respectivement le modèle sémantique et le graphe des réseaux places/transitions. Le troisième et le quatrième niveaux concernent respectivement le modèle sémantique et le graphe des réseaux de haut niveau.

2.2 Deuxième partie : métamodèles et format d'échange

L'approche mise en œuvre par le deuxième volet de la norme prend le parti d'une conception d'abord de haut niveau, indépendamment de toute technologie de représentation concrète. Ainsi, à travers une définition basée sur des métamodèles en UML, une mise en correspondance sémantique avec les définitions formelles de la première partie est établie. Cette approche fournit ainsi dans un premier temps une syntaxe abstraite pour la représentation des types de réseaux de Petri places/transitions, symétriques et de haut niveau.

La première partie adopte une approche mathématique englobante puis restrictive pour la définition des types de réseaux de Petri mentionnés. Dans une logique d'extensibilité (futures extensions, nouveaux types, modularité et hiérarchie dans la troisième partie), la deuxième partie implémente une approche modulaire et incrémentale. Cette approche a l'avantage de mettre en évidence la relation syntaxique qui existe entre les types et favorise ainsi la réutilisation de définitions communes. Cette relation syntaxique est bien entendu validée par la compatibilité sémantique entre les types.

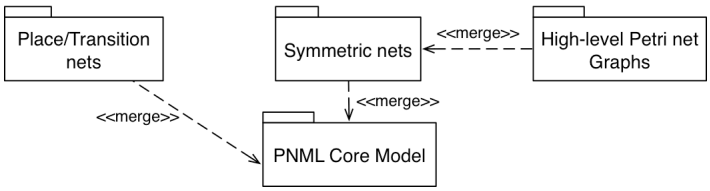

Figure 1. Packages des types de réseaux de Petri définis dans la deuxième partie et leurs relations.

La figure 1 illustre cette démarche, où le modèle noyau (*PNML Core Model*) représente tout graphe de réseau de Petri non spécifiquement étiqueté. Le seul type d'étiquette concrète concerne le nom des nœuds et des arcs. Dans ce graphe, les arcs relient des nœuds, sans restriction sur le type de nœuds connectés en entrée ou en sortie. Le modèle noyau ne désigne donc pas un type particulier de réseau de Petri. Il est présenté en figure 2, sans la partie définissant les aspects graphiques des nœuds et des étiquettes. Sur la figure 1, les réseaux places/transitions réutilisent le graphe du modèle noyau et définissent les types d'étiquettes autorisées (entiers pour le marquage des places et la valuation des jetons sur les arcs). Ils sont présentés en figure 3.

Les réseaux symétriques réutilisent le graphe du modèle noyau et y ajoutent les types de données algébriques finis (et les opérations associées) permettant de définir les « annotations colorées » telles que déclaration de domaines, de variables, marquages des places, inscriptions sur les arcs et gardes sur les transitions. Ces types de données algébriques concernent les énumérations finies, cycliques, les intervalles finis d'entiers, les booléens, les multi-ensembles et les *black dots*. Dans le cas des réseaux symétriques, les partitions constituent un type structurant sur les types précédents, permettant de faire ressortir les classes d'équivalence que partagent leurs éléments.

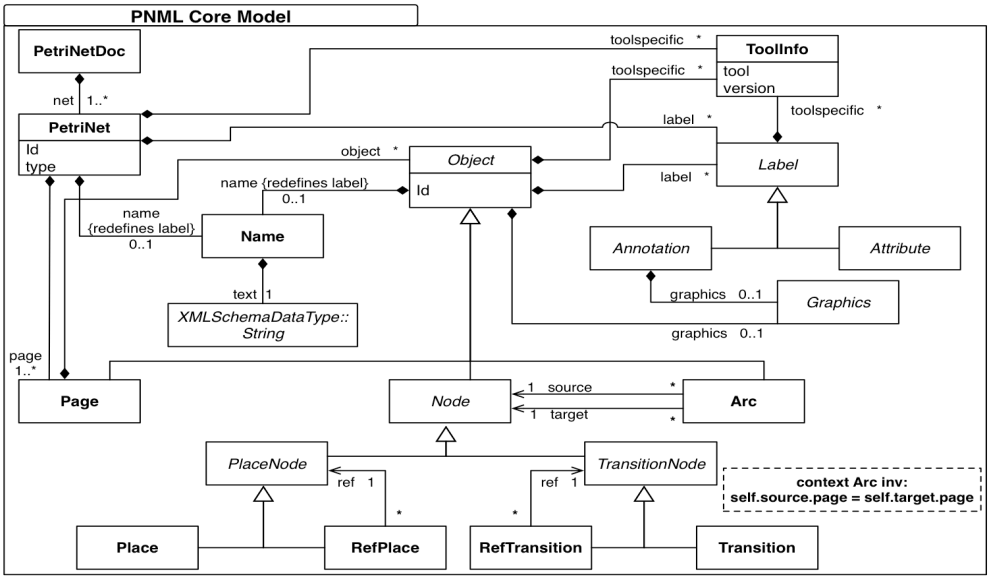

Figure 2. Modèle noyau, sans les aspects graphiques des nœuds et annotations

Enfin, les réseaux symétriques constituant une restriction des réseaux de haut niveau, ces derniers réutilisent la syntaxe des premiers. Les réseaux de haut niveau manipulent en plus des types infinis comme les entiers, les listes et les chaînes de caractères. De plus, des types arbitraires utilisateurs (et leurs opérations) peuvent être définis. Une vue abstraite de la construction des réseaux de haut niveau est présentée en figure 4, où l'on observe la réutilisation du package des réseaux symétriques.

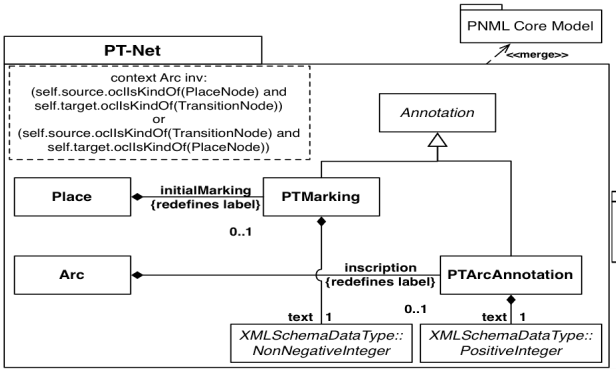

Figure 3. Réseaux places/transitions, réutilisent le noyau

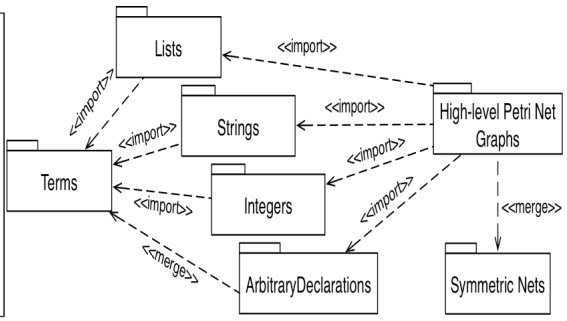

Figure 4. Réseaux de haut niveau

Une grammaire en XML dérivée des modèles de haut niveau présentés ci-dessus et les représentant, a ensuite été créée. Elle spécifie la syntaxe concrète d'échange des modèles de réseaux de Petri décrits selon les constructions de la norme. Ce langage est appelé Petri Net Markup Language (http://www.pnml.org).

3 Implémentation et dissémination

L'approche adoptée dans le développement de la seconde partie du standard a été expérimentée pour les réseaux places/transitions et symétriques au travers d'un ensemble d'interfaces de programmation (API) en Java, générées automatiquement à partir des métamodèles de la norme implémentés en Eclipse Modeling Framework (EMF). La bibliothèque ainsi produite fournit toutes les fonctions d'import et d'export de modèles en PNML nécessaires à un développeur d'outil, sans requérir de sa part une connaissance approfondie du standard. Le cadre générateur de ces API est un outil libre et gratuit, PNML Framework (http://pnml.lip6.fr).

3.1 PNML Framework

PNML Framework fournit une API de haut niveau constituée de méthodes de création de nœuds et de leurs annotations, en une invocation pour le développeur utilisateur, en plusieurs étapes enfouies (selon la structure du graphe, voir par exemple le modèle noyau) pour PNML Framework qui gère la complexité de la construction.

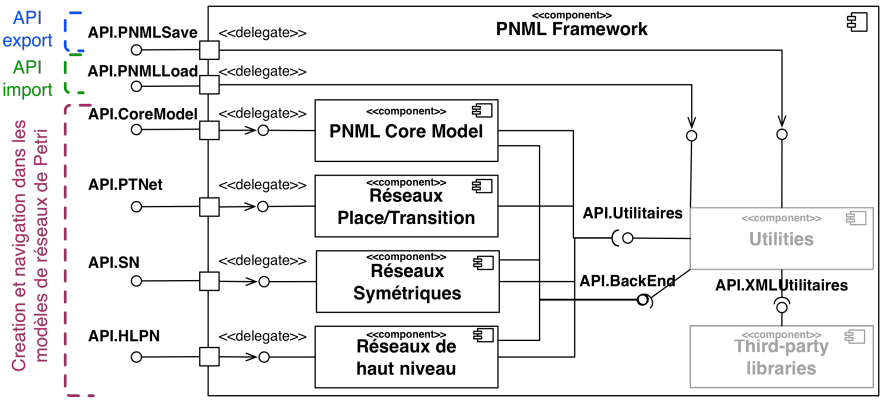

Figure 5. Architecture de PNML Framework

L'architecture de PNML Framework, présentée en figure 5, est conçue selon les techniques d'ingénierie dirigée par les modèles, en s'appuyant sur EMF. Le code des composants représentant chaque type de réseaux de Petri est entièrement généré. Le code manuel est localisé dans les utilitaires qui permettent de provoquer le chargement et la sauvegarde des fichiers PNML d'une part (*Utilities*), la lecture pas à pas d'un arbre XML d'autre part (*Third-party Libraries*). Ce code manuel se chiffre à 3600 lignes, ce qui représente moins de 1% du code engendré (les API natifs de EMF et les API spécifiques à PNML).

```
ModelRepository.getInstance().createDocumentWorkspace("coqWksp");                    1
PnmlImport pim = new PnmlImport();                                                    2
pim.setFallUse(true);                                                                 3
HLAPIRootClass imported = (HLAPIRootClass) pim.importFile("pnmlDocument");            4
Processor proc = MainProcessor.getProcessor(imported);                                5
    if (imported.getClass().equals(
    fr.lip6.move.pnml.ptnet.hlapi.PetriNetDocHLAPI.class))                            (5)
    { p = new PTProcessor(); return p;}
proc.process(imported, new PrintWriter(new FileWriter("coqDocument")));               6
```

Figure 6. Principales étapes pour importer un document PNML

L'utilisation de PNML Framework est en pratique assez simple. La figure 6 montre l'exemple du processus d'importation d'un document PNML par une application de transformation vers le format de l'outil Coq. Les six étapes de la figure 6 sont décrites ci-après :

1. Création d'un espace de travail dans PNML Framework afin d'y manipuler les modèles chargés.
2. Création d'un importateur de document PNML.
3. Si le type de réseau n'est pas standard, alors rétrograder au type normalisé le plus proche.
4. Importer le document PNML (par PNML Framework).
5. Une instance du moteur de traitement des documents PNML, écrit par le développeur-utilisateur de PNML Framework, est créée. Le développeur-utilisateur a déterminé (5) le type de réseau chargé et renvoyé la bonne instance du moteur.
6. Le moteur est invoqué pour réaliser la transformation vers le format de Coq.

3.2 Publications et séminaires

Nos travaux sur la norme comportent également un volet relatif à sa dissémination, en plus de l'aspect logiciel représenté par PNML Framework. Nous avons dans ce cadre publié des articles dans la revue Petri Net Newsletter n°69 et 76 [4,5], aux conférences internationales FORTE 2006 [6], CPN 2009 [5] et Petri nets 2010 [7]. Nous animons également régulièrement séminaires et tutoriels sur la norme en marge de la conférence annuelle Petri nets.

Nous sommes également en train de développer une application web, devant servir d'entrepôt de modèles de réseaux de Petri en PNML. L'interaction à distance avec cet entrepôt directement à partir d'outils aussi bien que via un navigateur classique est possible grâce à la technologie *Representational State Transfer* (REST). De plus, une API en Java implémentant cette interaction et intégrable dans les outils compatibles, a été développée.

Ces événements permettent d'une part de montrer à la communauté l'avancement des travaux et d'autre part de recueillir des commentaires nous permettant de progresser vers un standard accepté

4 Enjeux pour le futur

La troisième partie de la norme concerne les extensions. Les premiers travaux ont permis de cerner les problématiques sur lesquelles les efforts vont se concentrer. Plusieurs axes principaux sont à l'étude : la structuration des réseaux de Petri, leurs extensions et l'échange de propriétés ; pour les deux premiers, les travaux ont été entamés.

La structuration des réseaux de Petri vise à les équiper des concepts modulaires. Pour cela, plusieurs approches de la littérature ont été envisagées, conduisant à en retenir une et l'améliorer avec les aspects intéressants des autres. Ce travail, publié à Petri nets 2009 [8], distingue pour chaque module, une partie « interface » du module (réseau de Petri) lui-même. L'interface regroupe une vue des éléments qui sont connus de l'environnement, c'est-à-dire des nœuds ou des déclarations soit « importés » par le module depuis son environnement, soit «exportés » et dans ce cas fournis par le module. L'approche retenue présente en particulier l'avantage de pouvoir utiliser plusieurs instances d'un même module, de paramétrer des déclarations, assurant ainsi une grande flexibilité. Toutefois, certains enjeux restent à étudier, tels que la politique de composition. Par exemple, lorsque l'on synchronise plusieurs transitions, le passage de paramètres dans les réseaux de Petri de haut niveau peut correspondre à un échange de valeurs, l'opération peut traduire un rendez-vous multiple ou non, etc.

Les extensions de réseaux de Petri se déclinent de deux manières : des ajouts simples tels que les arcs inhibiteurs, ou la prise en compte d'un modèle enrichi comme les réseaux temporels. L'objectif est alors de permettre à l'utilisateur d'un modèle étendu de décrire les règles le régissant par un métamodèle à partir de ceux existant déjà. Cette nouvelle description devra bien sûr être compatible avec les travaux antérieurs pour ne pas les dénaturer, et pourra à terme, après une phase d'étude à la fois théorique et pratique, être intégrée au standard. Le mécanisme mis en œuvre pour créer ces nouvelles extensions fournira un cadre rigoureux.

5 Conclusion

ISO/IEC-15909 représente la concrétisation de nombreuses années d'efforts (Wheeler 1993, Bause 1995, Koelmans 1995, Berthelot 1998, Lyngso 1998, Jungler 2000, Sy 2000, Mailund 2000, Stehno 2002, Weber 2003 et Billington 2003) de la communauté travaillant sur les réseaux de Petri pour mettre en place un cadre formel et unifié de définitions et techniques de représentation, afin favoriser l'échange non ambigu de spécifications.

L'objectif fondamental d'interopérabilité entre outils devrait reposer au moins sur deux aspects importants : l'échange non ambigu de modèles et de résultats de vérification. Actuellement, seul le premier aspect est en cours de définition. Les travaux avancent généralement bien et ont donné lieu à une première version normalisée du format d'échange, prenant en charge les réseaux places/transitions, symétriques et de haut niveau. La prochaine version intégrera les extensions aux types existants et établira le cadre pour la définition de nouveaux types. La modularité et la hiérarchie feront également partie de cette prochaine version, qui est en cours d'élaboration dans la troisième partie de la norme.

6 Références bibliographiques

[1] M. DIAZ, éditeur : « Petri Nets, Fundamental Models », *Verification et Applications*, Wiley-ISTE, 2009

[2] C. GIRAULT et R. VALK : « Petri Nets for Systems Engineering », Springer Verlag – ISBN : 3-540-41217-4, 2003.

[3] K. JENSEN : « Coloured Petri Nets: Status and Outlook », in W. M. P. van der Aalst and E. Best, éditeurs, *ICATPN*, volume 2679 de *LNCS*, pages 1–2, Springer, 2003.

[4] L. HILLAH, F.KORDON, L. PETRUCCI et N. TREVES : « Model engineering on Petri nets for ISO/IEC 15909-2: API

Framework for Petri Net types metamodels », *Petri Net Newsletter*, (69):22–40, 2005.

[5] L. HILLAH, E. KINDLER, F. KORDON, L. PETRUCCI et N. TREVES : « A primer on the Petri Net Markup Language and ISO/IEC 15909-2 », *Petri Net Newsletter* (initialement présenté au *10th International workshop on Practical Use of Colored Petri Nets and the CPN Tools* – CPN'09), 76:9–28, 2009.

[6] L. HILLAH, F. KORDON, L. PETRUCCI et N. TREVES : « PN standardisation : a survey », in *International Conference on Formal Methods for Networked and Distributed Systems* (FORTE'06), pages 307–322, 2006.

[7] L. HILLAH, F. KORDON, L. PETRUCCI et N. TREVES : « PNML Framework: an extendable reference implementation of the Petri Net Markup Language », in *Petri Nets 2010*, volume à paraître, Springer, 2010.

[8] E. KINDLER et L. PETRUCCI : « Towards a Standard for Modular Petri Nets: A Formalisation », in *Application and Theory of Petri Nets*, volume 5606 de *LNCS*, pages 43–62, 2009.