

Organizational Petri Nets for Protocol Design and Enactment

Stéphanie Combettes
University Toulouse 3
115, Route de Narbonne
31077 Toulouse, France
stephanie.combettes@iut-
tlse3.fr

Chihab Hanachi
IRIT - University Toulouse 1
Place Anatole France
31042 Toulouse, France
hanachi@univ-tlse1.fr

Christophe Sibertin-Blanc
IRIT - University Toulouse 1
Place Anatole France
31042 Toulouse, France
sibertin@univ-tlse1.fr

ABSTRACT

To integrate protocols in real organizations, we need to take into account three complementary dimensions: the protocols behavior, the structure and deontic aspects of the organization being considered and its informational model. Basic Petri Nets (PN) are known to deal suitably with the behavioral aspect while some high level extensions are intended to take into account one of the additional dimensions. This paper proposes a formalism called *Organizational Petri nets* to capture these three dimensions in a coherent framework. We give the definition of this new formalism, precise its semantics and illustrate its use through an example.

Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques – *Petri Nets, Object-Oriented design Methods*

D.3.1 [Programming languages]: Formal Definitions and Theory – *Semantics, syntax*.

General Terms

Design, Theory.

Keywords

Petri Net, deontic aspect, protocol, agent.

1. INTRODUCTION

Protocols are one of the basic building blocks [7] for structuring the design of Multi-Agent Systems (MAS) and coordinating their execution. In real-world applications such as Agent-Oriented Information Systems (AOIS), protocols involve three inter-dependant dimensions: the behavioral, organizational and informational ones. These three dimensions may capture by the three inter-related models described below. *The behavioral model* describes the protocol control structure, i.e. the actions (performatives sending or receiving, elementary actions) constituting the steps of a protocol, and their coordination (a pre-order relation between these interventions). *The organizational model* should refer to the logical actors able to carry out the activity of the organization, in order to define the specific contribution of each role with regard to the others and the rule applied to grant these roles to the concrete actors. This determines "what agents have permission to do within a protocol". Moreover, AOIS applications – such as e-alliance, inter-organizational

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

workflow or e-institutions – often deal with documents conform to regulations like contracts or norms. In this case two other deontic aspects should be represented: obligation and prohibition. Finally, the *informational model* represents documents, forms and data having attributes that are handled and exchanged by the actors of the organization, possibly through the contents of communicative acts.

Although High Level Petri Nets are suitable for specifying, validating, simulating and even implementing protocols (see [1],[2],[3]), when referring to the state of the art, no Petri net dialect meets all the previous requirements. However, two works merit to be examined. The first one is due to Raskin and al. [9] who extend Petri nets by integrating deontic aspects. Unfortunately, this work is based on basic Petri nets and does not allow to represent the actors and information involved in Protocols. In other words, protocols cannot be described within their execution context. The second work, by Sibertin [5], proposes the Petri Nets with Objects (PNO) formalism which combines coherently the Petri net theory and the Object-Oriented (OO) approach. While PN are very suitable to express the dynamic behavior of protocols, the OO approach permits the modeling and structuring of its active (actors) and passive (information resources) entities. In a basic PN, tokens are atomic undistinguishable items, whereas in a PNO they are objects provided with attributes and methods. Unfortunately, PNO do not take into account the deontic aspect of Protocols. Notably, they do not give means to represent the obliged and prohibited actions.

Consequently, the aim of this paper is to provide a High Level Petri net language integrating the above three dimensions (behavioral, organizational and informational) to represent AOIS protocols. The proposed formalism is a combination of the two above-mentioned languages: Deontic Petri Nets [4] and PNO [5]. After presenting the formalism, we develop an example for the modeling of the book borrowing procedure of a library.

2. ORGANIZATIONAL PETRI NETS

2.1 Syntactic Definition of OgPN

Definition 1 (OgPN): An Organizational Petri Net (OgPN) is a n-uple $(C, P, T, V, PreCd, Act, EmR, Pre, Post, Resp, Cost, CTD)$:

- C is the set of classes of the object processed by the net, $C = A \cup D$ where A contains the classes of active objects (the organizational actors) and D contains the classes of passive data or resources; any object class in A is assumed to have a predefined *penalty* attribute;
- P is the set of *places*, typed by a function $P \rightarrow C^*$;

- T is the set of *transitions*, $T = Id \cup Vi \cup Re$ where these subsets are respectively the sets of *ideal*, *violation* and *repair* transitions. Transitions in Vi and Re are the sub-ideal ones;
- V is the set of *variables* that label the arcs;
- $PreCd$ is the set of *preconditions* that guard transitions;
- Act is the set of *actions* performed on transition occurrences;
- EmR is the set of *emission rules* that control the direction of tokens at the completion of transitions;
- Pre is the *forward incidence function*: $P \times T \rightarrow V^*$;
- $Post$ is the *backward incidence function*: $T \times EmR \times P \rightarrow V^*$;
- $Resp$ is the function that associates to each sub-ideal transition a variable corresponding to the *responsible actor*: $Vi \cup Re \rightarrow V$ where the type of $Resp(t)$ belongs to A ;
- $Cost$ is the function that states the punishment or reward resulting from the occurrence of a transition t ; if $t \in Id$ then $Cost(t) = 0$; if $t \in Vi$ then $Cost(t) > 0$; if $t \in Re$ then $Cost(t) < 0$;
- $CTD \subset Vi \times Re$, is a relation which associates each repair transition to violation transitions and thus defines the *Contrary To Duty* obligations: $(t, t') \in CTD \Leftrightarrow t'$ repairs t .

Each sub-ideal transition must have an actor responsible for that transition, which will assume the cost of performing this transition. Accordingly, a violation or repair transition has an input variable whose type belongs to A , and at each occurrence of the transition, the actor linked to that variable is charged with the value of the *Cost* function. The *penalty* attribute of the object classes in A is intended to store the amount of each actor penalties.

2.2 Semantics of Organizational Petri Nets

The *Organizational Petri Nets* formalism is a combination of the *Deontic Petri Net* and *Petri Net with Object* formalisms, which both have a well-funded semantics extending the *basic Petri nets* semantics. In this line, the OgPN semantics includes the PNO semantics (as defined in [6]) and introduces into actor objects the DPN's computation of penalties.

A transition may occur, by a *substitution principle* [5], only if it can pick tokens in its input places in such a way that each of its input variables is bound to an element of its domain of value.

Definition 2 (Transition firing): The occurrence of a transition t that may occur with a substitution S from a marking M yields the marking M' (noted $M \xrightarrow{t, S} M'$) by following this 5-steps process:

1. Withdraw from the input places of t the tokens bound to its input variables: $M'(p) := M(p) \setminus S(Pre(p, t))$;
2. Generate new objects, if needed to instantiate the output variables of t that are not already input variables;
3. Perform the action Act_t of t , that most often consists in calling methods or updating values of attributes of the objects bound to the transition variables;
4. If the transition t is a violation or repair one, update the penalty of the actor responsible for the occurrence of t :
a.penalty := a.penalty + $Cost(t)$, where $a = S(Resp(t))$;
5. Evaluate emission rules EmR_t of t to decide the one r that is satisfied, and put tokens bound to variables into the appropriate output places: $M'(p) := M(p) + S(Post(t, r, p))$.

Steps 1 and 5 define the location of tokens into places, while steps 2 to 4 compute the value of these tokens.

The **cost** associated to the executions of a transition is a penalty of the actor responsible for that transition occurrence. Provided that its initial value is null, the penalty attribute of an actor is the sum of the costs of the transition occurrences he is responsible for. Most often, the execution of a sequence of transitions is more significant than the occurrence of a single transition in isolation. Let $s = t_1. \dots .t_n \in T^*$ be a sequence of transitions and M a marking. s is enabled from M and its occurrence yields the marking M' if substitutions $S_1, \dots, S_n$ and marking $M_1, \dots, M_n = M'$ such that $M \xrightarrow{t_1, S_1} M_1 \xrightarrow{t_2, S_2} \dots \xrightarrow{t_n, S_n} M_n$ exist.

According to the semantics of DPN, the **cost of a transition sequence** $s = t_1. \dots .t_n$ is the sum of the costs associated to the transitions in s : $Cost(s) = \sum_{i=1..n} Cost(t_i)$. If an actor a is responsible for all the transitions of a sequence s , the performance of this sequence will update his penalty in the following way: a.penalty := a.penalty + $Cost(s)$.

3. EXAMPLE

Let us first describe the protocol for borrowing books from a library. Someone, who has borrowed a book, must return it in time and without damages. If the borrower delays the book delivery he is penalized. Likewise, if he damages the book, he is first penalized but he may repair the damage. In case of repairing, his penalty is reduced.

The design of an OgPN consists in three inter-dependant stages that correspond to the production of three models: informational, organizational and behavioral models.

The *informational model* describes the structure of the information together with some integrity constraints. In an OgPN, information units are represented as object classes, using the powerful mechanisms (such as classification, inheritance, polymorphism...), that allow to describe complex data structure (forms, rules...) usually used in normative systems. These objects will flow in the OgPN and are transformed by actions performed by the actors. In our example, we only consider the attributes of the Book class (see table 1).

Table 1. The Book Class

Class Book
<u>Attributes:</u>
Title: string
Author: string
Reference: string
ReturnDate: date
Aspect: {Good, Damaged, Repaired}
State: {Available, Returned, Borrowed}
End

The *organizational model* structures the actors in object classes sharing the same features. A class represents a role and can comprise actors having the same capabilities or belonging to the same organizational unit. It also attributes to each actor, permission to perform actions on objects of the informational model and the responsibility of some actions that violate the rules of the organization. In our context, roles are described as objects or agent classes; the permissions are implicitly given by the declaration of method's parameters, which refer the objects classes of the informational model. The inheritance mechanism propagates permissions between roles. An actor is responsible for

all actions implemented by methods, it can perform, and its "penalty" attribute records its amount of penalties. The organizational model is limited in our example to a single role described by the Borrower class (see table 2). This role contains attributes and methods, and has the permission to handle books since this type of object appears as a parameter of the methods. The penalty attribute is updated each time an agent playing the role transgresses a deontic rule expressed in the Petri net.

Table 2. The Borrower Class

Class Borrower	
<u>Attributes:</u>	
Name: string	
Penalty: integer	
<u>Methods:</u>	
Borrow (B : Book)	% triggered when the borrower borrows the book
Read (B : Book)	% triggered when he reads the book
ReturnLate (B : Book)	% triggered when he returns the book late
Return (B : Book)	% triggered when he returns the book
Repair (B : Book)	% triggered when he repairs the damaged book
Damage (B : Book)	% triggered when he damages the book
End	

Finally, the *behavioral model* describes how the protocols behave by defining the action sequences and the information and roles involved in each action. Thanks to the high expressive power of Petri nets, the designer may express a lot of control structures (parallelism, sequence, alternative...), which represent deontic constraints between two actions. The permissions granted to a role are expressed by the nature of the arc between an information place and a transition. The ideal, violation and repair actions are represented by transitions with their respective cost. Violation and repair transitions charge the actor responsible for that transition.

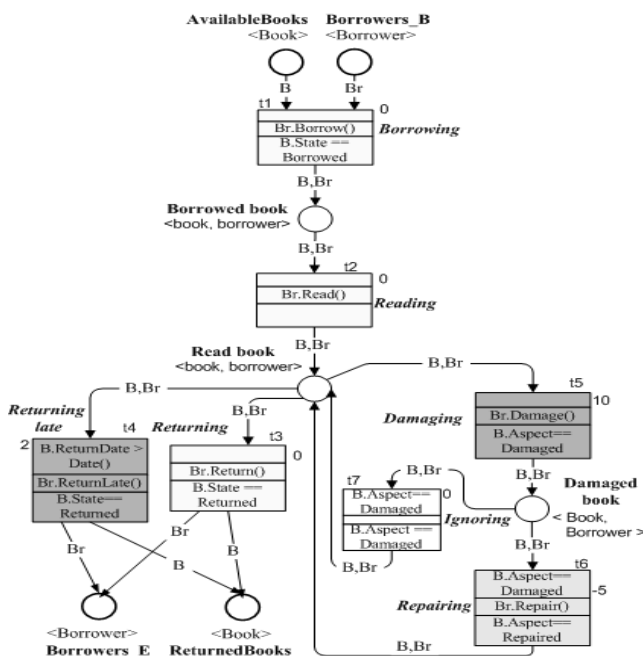


Figure 1. OgPN modeling borrower behavior

Regarding the behavioral model (see figure 1), the organizational Petri net includes seven transitions and six places typed with the

two previously defined classes. The two initial places (*AvailableBooks* and *Borrowers_B*) represent a starting situation where a Borrower arrives in order to borrow an available Book. The two final places (*ReturnedBooks* and *Borrowers_E*) correspond to a situation where the Borrower has returned the Book. The transitions are grayed to distinguish more easily between ideal transitions (white: t1, t2, t3, t7), transgression transitions (dark grey: t4, t5) and repair transitions (bright grey: t6). The number associated to each transition represents its cost. For example, the transition t6 called *Repairing* is a repair transition and its cost is "-5". Here the responsible actor is implicit because there is only one actor. However in case of ambiguity, the cost is prefixed with the responsible class name: <name>: cost. Let us explain a scenario represented by sequence s2 = t1.t2.t5.t6.t4 corresponding to the case where the Borrower borrows the Book (t1), reads it (t2), damages it (t5), repairs it (t6) and finally returns it late (t4). The penalty gained by the Borrower agent linked to the Br variable during an execution of this sequence is the sum of the costs of the transitions, that is $10 - 5 + 2 = 7$.

Let us notice that this OgPN is an executable specification that can be used for simulation purpose. To completely control the operational functioning of a library, the code of the methods of object classes must be given.

4. CONCLUSIONS

We gave the formal semantics of the formalism called *Organizational Petri Nets* and some guidelines to produce models through an example. The advantages of this formalism are: 1) an easy integration in real world organization 2) the possibility of simulating protocols before their deployment 3) behavioral analysis capabilities 4) an operational semantics. In future work, we will investigate how this formalism can help to support the notion of protocol in existing agent-oriented methodologies like GAIA.

5. REFERENCES

- [1] Cost R.S., Chen Y., Finin T., Labrou Y., Peng Y., *Modeling Agent Conversations with Colored Petri Nets*. Workshop on Specifying and Implementing Conversation Policies, Seattle, 1999, pp. 59-66.
- [2] Hanachi C., Sibertin-Blanc C., *Protocol Moderators as active Middle-Agents in Multi-Agent Systems*. Journal of AAMAS, 8, 131-164, 2004.
- [3] Mazouzi H., EL Fallah-Seghrouchni A., Haddad S., *Open Protocol Design for complex Interactions in Multi-agent Systems*. First International Joint Conference on AAMAS, Italy, 2002.
- [4] Raskin J.F., Tan Y.H. and van der Torre L.W.N. *Modeling Deontic States in Petri Nets*. Seventh Dutch AI Conference (NAIC-95), the Netherlands, 1995.
- [5] Sibertin-Blanc C. *High Level Petri Nets with Data Structure*. 6th European Workshop on Application and Theory of Petri Nets, p141 - 170, 1985.
- [6] Sibertin-Blanc C. *A functional semantics of Petri Nets with Objects*. IRT Report, University of Toulouse 1, 1992.
- [7] Sturm A., Shehory O. *A Framework for Evaluating Agent-Oriented Methodologies*. Agent Oriented Information System Workshop, 2003: 94-109.