Formal Composition Based on Roles within a Model Driven Engineering Approach

Cédrick Lelionnais
Jérôme Delatour
and Matthias Brun
ESEO-TRAME
Angers, FRANCE
{cedrick.lelionnais,jerome.delatour,matthias.brun}@eseo.fr

Olivier H. Roux
and Charlotte Seidner
IRCCyN - Université de Nantes
École Centrale de Nantes
Nantes, FRANCE
{olivier-h.roux,charlotte.seidner}@irccyn.ec-nantes.fr

***Abstract*—Faced with the increasing complexity of Real-Time Embedded Systems, Model Driven Engineering offers the possibility of developping frameworks in which transformations are used to generate either executable code or formal models. However, these transformations themselves are generally not formalized. Correctness of transformations could therefore be called into question. This paper proposes a formalization of a transformation step, namely: the composition of formal fragments describing the behavior of a real-time system. These fragments are described using an extension of the classical Time Petri Nets, where the notion of roles was added to perform the composition of the fragments. This formalization increases confidence in transformations.**

***Keywords*—Model Driven Engineering, Real-time operating systems, Behavioral modeling, Transformation, Verification, Time Petri Nets, Application deployment**

I. Introduction

Real-Time Embedded Systems (RTESs) increasingly surround us in various domains (aircrafts, automotive sector, cell phones, robotics, etc.). RTES engineers are confronted with the challenge of developing more complex, higher quality systems, with shorter development cycles at lower costs. Model Driven Engineering (MDE) [7] helps engineers to develop frameworks for partially automating the development of RTESs. Thanks to transformations, those frameworks produce either executable code or formal models from high-level descriptions of RTESs.

However, many frameworks do not consider the description of Real-Time Operating Systems (RTOSs) [4]. RTOSs have indeed an impact on the behavior of RTESs. In addition, in spite of the fact that the behavior of RTOSs starts to be considered, transformations have often been implemented within frameworks without formalization. Correctness of the transformation could therefore be called into question. Confidence in those frameworks could also be reduced.

The general approach presented in this paper aims to create a formal model of the whole system deployed on a RTOS. This approach was thought regardless of the intended RTOS. To do this, a transformation process is currently in progress to compose several behavioral fragments, each one describing a part of this system. Those fragments come from a model of the targeted RTOS, which is considered through the process execution. However, composition rules must be chained in a right sequence in order to avoid any ambiguity. As a basis of the construction, the use of roles formally identify connection points, which will be used as a glue of the system parts.

This paper is divided into the following sections. Section 2 refers briefly to the frameworks chosen for this contribution. The latter is presented in Section 3, highlighting both the deployment process and the use of roles. Section 4 deals with Time Petri Nets (TPNs) as translation formalism. A new syntax is then defined formalizing the composition of TPNs based on roles. Relying on this definition, Section 5 formalizes the construction of an application deployment in TPN. Consequently, the benefits and limits of this approach are discussed in Section 6. Finally, we conclude in Section 7.

II. Related Works

A first presentation of related works in conjunction with the consideration of RTOSs has already been presented in a previous contribution [4]. For this reason, frameworks in line with the consideration of RTOSs will only be presented here.

We have opted for frameworks in which the intervention of each stakeholder has been made more flexible. Indeed, the domain skills (RTOSs structure, transformations, deployments choice, etc.) are correctly separated with these frameworks. This has been made possible thanks to an explicit approach, which consists in considering each RTOS description without modifying the transformation rules. This strategy offers the possibility to capitalize most descriptions in a generic way. Furthermore, other works [3] [4] are based on the behavioral consideration of RTOSs. These contributions search for refining models of applications deployed on RTOS with the aim of verifying properties.

We can note the MARTE UML profile [8] in which the Software Resource Modeling (SRM) approach [12] has been integrated. With SRM, RTOSs can be modeled using stereotyped concepts from the real-time software domain. For another example, Software Execution Platform Inside Tools (SExPIsTools) is involved in the tooling of development processes. Real-Time Embedded Platform Modeling Language (RTEPML) [2] was developped in this sense, with the aim of defining concepts dedicated to the real-time domain for modeling RTOSs.

However, as introduced previously, the processes implemented in those frameworks have not yet been formalized. We have therefore decided to carry on developping SExPIsTools by experimenting the formalization.

III. Contextualization in a MDE Approach

This section is divided into two parts. The first part presents the SExPIsTools framework and the language RTEPML [2]. This presentation details the notion of role, which is used for the composition of behavioral descriptions. The second part specifies the language adopted for formalizing.

 ISBN: 978-1-61208-307-0

A. SExPIsTools Framework

SExPIsTools (Figure 1) allows to generate code from high-level descriptions [2] to several RTOSs. The RTOS description (i.e., the Platform Description Model) is a parameter of the generic transformation made possible thanks to the notion of role. A role explicitly establishes a relationship between abstract concepts of RTOSs (i.e., the notion of task), their properties (priority of task) and their services (creation or destruction of task). The transformation rules rely on both concepts and roles. For each Platform Description Model, translation of roles is given in the Application Programming Interface (API) of the targeted RTOS. The descriptions are realized by the modeling language RTEPML.

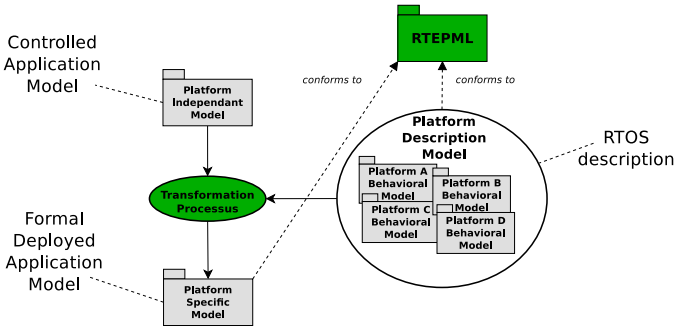

Fig. 1: SExPIsTools Process within MDE Context

RTEPML has been extended [4] to describe RTOSs behavior in a formal way. The purpose of formalizing was to allow model-checking. For each concept and service of the RTOS, a formal description (called fragment) is given. The transformation process leads to the composition of fragments. To facilitate this composition, some roles were added.

B. Choice of formal language

We have chosen TPNs [6] [1] to translate behavioral fragments because we need a formalism which expresses models with synchronism and parallelism for multitasking. Lastly, RTOSs imply time constraints. The chosen formalism needs to have clocks to represent time evolution.

IV. TPN Composition based on Roles

In order to compose fragments in TPN, we have projected roles on those fragments. To perform the composition, we have decided to assign roles to places. The interest of such a method is to merge places [11] [10], which are the connection points of the system that must be modeled in TPN.

In this section, TPNs with roles are firstly defined. The definition of the instanciation of TPN with roles is then given. Finally, the composition of TPNs is highlighted through a synchronization formalism based on roles.

A. TPNs

TPNs are a timed extension of classical Petri nets. Informally, to each transition of the net is associated an implicit clock and an explicit time interval. The clock measures the time since the transition has been enabled and the time interval is interpreted as a firing condition: the transition may fire if the value of its clock belongs to the time interval.

Definition 1 (TPN): A TPN is a tuple $\mathcal{T} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, I_s \rangle$ where:

- P is a finite non-empty set of *places*;
- T is a finite non-empty set of *transitions*;
- $\mathsf{Pre} : P \times T \to \mathbb{N}$ is the *backward incidence* function;
- $\mathsf{Post} : P \times T \to \mathbb{N}$ is the *forward incidence* function;
- $m_0 : P \to \mathbb{N}$ is the *initial marking* of the net;
- $I_s : T \to \mathbb{N} \times (\mathbb{N} \cup \{+\infty\})$ assigns a *static time interval* to each transition.

A marking of $\mathcal{T}$ is an application from P to $\mathbb{N}$. Let m be a marking of $\mathcal{T}$. Then, for any place $p \in P$, we say that p contains $m(p)$ *tokens*. A transition $t \in T$ is said to be enabled by the marking m if $\forall p \in P, m(p) \geq \mathsf{Pre}(p,t)$. This is denoted by $t \in \textit{enabled}\,(m)$. For any interval I_s, we denote by $I_s{}^{\downarrow}$ the smallest left-closed interval with lower bound 0 that contains I_s. For each transition t there is an associated clock x_t. We consider valuations on the set of clocks $\{x_t | t \in T\}$ and we will slightly abuse the notations by writing $v(t)$ instead of $v(x_t)$.

Let m be a marking of the net and t a transition in $\textit{enabled}\,(m)$. Let m' be the marking obtained from m by firing t. Let m'' be the *intermediate marking* defined by $\forall p, m''(p) = m(p) - \mathsf{Pre}(p,t)$. A transition t' is *newly enabled* by the firing of t from m, and we note $t \in \uparrow\textit{enabled}\,(m,t)$ if $t' \in \textit{enabled}\,(m') \setminus \textit{enabled}\,(m'') \cup \{t\}$

The operational semantics of the TPN $\mathcal{T} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, I_s \rangle$ is defined by the time transition system $\mathcal{S}_\mathcal{T} = (Q, q_0, \to)$ such that:

- $Q = \mathbb{N}^P \times \mathbb{R}^T_{\geq 0}$
- $q_0 = (m_0, \mathbf{0})$
- $\to \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relation including a discrete transition and a continuous transition.
  - The discrete transition is defined $\forall t \in T$ by $(m,v) \xrightarrow{t \in T} (m',v')$ iff:
    - $t \in \textit{enabled}\,(m)$;
    - $\forall p \in P, m'(p) = m(p) - \mathsf{Pre}(p,t) + \mathsf{Post}(p,t)$;
    - $v(t) \in I_s(t)$;
    - $\forall k \in [1, |T|], \quad v'_k(t_k) = \begin{cases} 0 \textbf{ if } t_k \in \uparrow\textit{enabled}\,(m,t) \\ v_k(t_k) \textbf{ otherwise} \end{cases}$
  - The continuous transition is defined by $(m,v) \xrightarrow{d \in \mathbb{R}_{\geq 0}} (m, v+d)$ iff $\forall t' \in \textit{enabled}\,(m), \forall 0 < d' \leq d, (v+d')(t') \in I_s^{\downarrow}(t')$.

Definition 2 (TPN with roles): A TPN with roles is a tuple $\mathcal{N} = \langle \mathcal{T}, R, \lambda \rangle$ where:

- $\mathcal{T}$ is a TPN,
- R is a finite set of roles,
- $\lambda : P \to R \cup \{\perp\}$ is the function assigning a role to a place and $\perp$ denoting that no role is assigned to a

place. Hereafter, some notations and properties of this function are enumerated :

1) $P_\lambda = \{p \in P \mid \lambda(p) \neq \bot\}$ is the set of places with role.
2) $\lambda_{\setminus P_\lambda} : P_\lambda \to R$ is an injective function;
3) $\lambda^{-1} : R \cup \{\bot\} \to P \cup \{\emptyset\}$ such that
$$\begin{cases} \forall r \in R,\ \lambda^{-1}(r) = \begin{cases} p \text{ if } \lambda(p) = r \\ \emptyset \text{ otherwise} \end{cases} \\ \lambda^{-1}(\bot) = \emptyset \end{cases}$$

The operational semantics of the TPN with roles $\mathcal{N} = \langle \mathcal{T}, R, \lambda \rangle$ is the same as that of TPN. Indeed, the use of roles within the definition of TPN does not impact its semantics.

B. Instanciation of TPN with roles

As seen previously, some fragments of TPN are instanciated before being composed. In order to distinguish the fragments to compose, all roles in a same fragment must be renamed according to the name of the instance.

Let $\mathcal{N}$ be the TPN to instanciate and x the label given to the instance. The renaming function $\rightharpoonup$ is a function from R to R_i where assigned roles are involved in parameters.

Definition 3 (Instanciation of TPN with roles): The instanciation of $\mathcal{N}$ with m renamings is denoted by

$$\mathcal{N}_i = Ins(\mathcal{N}, x) = \mathcal{N} \Big|_{\substack{r^1 \rightharpoonup r^1\_x \\ \dots \\ r^m \rightharpoonup r^m\_x}}$$

with $m = |R|$, $\forall j \in [1,m]$, $r^j \in R$, $r^j\_x \in R_i$ and $\forall k \in [1,m]$, $k \neq j \to r^k \neq r^j$

C. TPNs Synchronization based on roles

In order to synchronize some TPNs, we must precise the definition of the composition of TPNs, which will be based on roles assigned to places. Let $\mathcal{N}_1, \dots, \mathcal{N}_n$ be n TPNs with $\mathcal{N}_i = \langle P_i, T_i, \mathsf{Pre}_i, \mathsf{Post}_i, m_{0_i}, I_{s_i}, R_i, \lambda_i \rangle$ such that $\forall k \neq k' \in [1,n] \implies T_k \cap T_{k'} = \emptyset$ and $P_k \cap P_{k'} = \emptyset$. The composition $\mathcal{N} = \langle P, T, \mathsf{Pre}, \mathsf{Post}, m_0, I_s, R, \lambda \rangle$ of the previous TPNs with roles will be denoted by $\mathcal{N} = \mathcal{N}_1 || \mathcal{N}_2 || \dots || \mathcal{N}_n$. Linked to this composition, we define a function leading to the merging of places whose assigned roles will be taken into account in parameters.

The merging function $\hookrightarrow$ is a partial function from $(R_1 \cup \{\bullet\}) \times (R_2 \cup \{\bullet\}) \times \dots \times (R_n \cup \{\bullet\}) \to P \times R$ where $\bullet$ is a special symbol used when a TPN is not involved in a particular merge of the global system. We then extend the definition of the assigning inverse function with $\lambda^{-1}(\bullet) = \emptyset$

The composition of n TPN with m merging is denoted by

$$\Big(\mathcal{N}_1 || \dots || \mathcal{N}_n\Big) \Big|_{\substack{(r_1^1, \dots, r_n^1) \hookrightarrow (p^1, r^1) \\ \dots \\ (r_1^m, \dots, r_n^m) \hookrightarrow (p^m, r^m)}}$$

with $\forall i \in [1,n]$, $\forall j \in [1,m]$, $r_i^j \in R_i$, $r^j \in R$ and $p^j \in P$, and $\forall k \in [1,m]$, $k \neq j \Rightarrow r_i^k \neq r_i^j$

We will subsequently use the following notations:

- Let $P_i^{merged} \subseteq P_i$ be the set of places of the net $\mathcal{N}_i$ merged by the composition. Formally $P_i^{merged} = \bigcup_{\forall j \in [1,m]} \{\lambda_i^{-1}(r_i^j)\}$
- Let $P^{\hookrightarrow} \subseteq P$ be the set of places of the net $\mathcal{N}$ obtained by the merging. Formally $P^{\hookrightarrow} = \bigcup_{\forall j \in [1,m]} \{p^j\}$

Definition 4 (Composition of TPNs with roles): The composition of the n TPN $\mathcal{N}_i$ with the merging $\hookrightarrow$ denoted by:

$$\mathcal{N} = \Big(\mathcal{N}_1 || \dots || \mathcal{N}_n\Big) \Big|_{\substack{(r_1^1, \dots, r_n^1) \hookrightarrow (p^1, r^1) \\ \dots \\ (r_1^m, \dots, r_n^m) \hookrightarrow (p^m, r^m)}}$$

is defined by:

- $R = \Big(\bigcup_{\forall i \in [1,n]} \big(R_i \setminus \bigcup_{\forall j \in [1,m]} \{r_i^j\}\big)\Big) \cup \Big(\bigcup_{\forall j \in [1,m]} \{r^j\}\Big)$;
- $P = \Big(\bigcup_{\forall i \in [1,n]} P_i \setminus P_i^{merged}\Big) \cup P^{\hookrightarrow}$;
- $T = \bigcup_{\forall i \in [1,n]} T_i$;
- $\lambda : P \to R$ is defined by:
  - $\forall p \in P \setminus P^{\hookrightarrow}$ meaning that $\exists i$ such that $p \in P_i$ then $\lambda(p) = \lambda_i(p)$
  - $\forall p^j \in P^{\hookrightarrow}$, meaning that p is the result of a merging, $\lambda(p^j) = r^j$
- $\mathsf{Pre} : P \times T \to \mathbb{N}$ is defined $\forall p \in P$ and $\forall t \in T_i \subseteq T$ by
$$\mathsf{Pre}(p,t) = \begin{cases} \mathsf{Pre}_i(p,t) \text{ if } p \in P \setminus P^{\hookrightarrow} \text{ and } p \in P_i \\ \mathsf{Pre}_i(p',t), \text{ if } \begin{cases} p \in P^{\hookrightarrow} \text{ and } p' \in P_i \\ (\dots, r_i^k, \dots) \hookrightarrow (p, \lambda(p)) \\ \lambda_i(p') = r_i^k \end{cases} \\ 0 \text{ otherwise.} \end{cases}$$
- $\mathsf{Post} : P \times T \to \mathbb{N}$ is defined $\forall p \in P$ and $\forall t \in T_i \subseteq T$ by
$$\mathsf{Post}(p,t) = \begin{cases} \mathsf{Post}_i(p,t) \text{ if } p \in P \setminus P^{\hookrightarrow} \text{ and } p \in P_i \\ \mathsf{Post}_i(p',t), \text{ if } \begin{cases} p \in P^{\hookrightarrow} \text{ and } p' \in P_i \\ (\dots, r_i^k, \dots) \hookrightarrow (p, \lambda(p)) \\ \lambda_i(p') = r_i^k \end{cases} \\ 0 \text{ otherwise.} \end{cases}$$
- $m_0 : P \to \mathbb{N}$ is defined $\forall p \in P$ by:
$$m_0(p) = \begin{cases} m_{0i}(p) \text{ if } p \in P \setminus P^{\hookrightarrow} \text{ and } p \in P_i \\ \sum_{i=1}^{n} m_{0i}\big(\lambda^{-1}(r_i^k)\big) \text{ if } \begin{cases} p \in P^{\hookrightarrow} \\ (r_1^k, \dots, r_n^k) \hookrightarrow (p, \lambda(p)) \end{cases} \end{cases}$$
- $I_s : T \to \mathcal{I}$ is defined $\forall t \in T$ by: $I_s(t) = I_{s_i}(t)$ **if** $t \in T_i$

As an example, $\mathcal{N} = \Big(\mathcal{N}_1 || \mathcal{N}_2 || \mathcal{N}_3\Big) \Big|_{(r_1, r_2, \bullet) \hookrightarrow (p, r)}$ is the parallel composition of the 3 TPNs, i.e., $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_3$, where the place $p_1 \in P_1$ such that $\lambda_1(p_1) = r_1$ and the place $p_2 \in P_2$ such that $\lambda_2(p_2) = r_2$ are merged. The name of the place obtained by this merging in $\mathcal{N}$ is $p \in P$ and its role is $\lambda(p) = r \in R$.

Property 1 (Associativity): The composition of TPNs with roles is associative in the following sense:

$$\Big(\mathcal{N}_1 || \mathcal{N}_2 || \mathcal{N}_3\Big) \Big|_{(r_1, r_2, r_3) \hookrightarrow (p, r)} = \Big(\big(\mathcal{N}_1 || \mathcal{N}_2\big) \Big|_{(r_1, r_2) \hookrightarrow (p_{12}, r_{12})} || \mathcal{N}_3\Big) \Big|_{(r_{12}, r_3) \hookrightarrow (p, r)}$$

$$= \left(\mathcal{N}_1 || \left(\mathcal{N}_2 || \mathcal{N}_3\right)\Big|_{(r_2,r_3)\hookrightarrow(p_{23},r_{23})}\right)\Big|_{(r_1,r_{23})\hookrightarrow(p,r)}$$

Property 2 (Commutativity): The composition of TPNs with roles is commutative:

$$\left(\mathcal{N}_1 || \mathcal{N}_2\right)\Big|_{\substack{(r_1^1,r_2^1)\hookrightarrow(p^1,r^1)\\ \cdots \\ (r_1^k,r_2^k)\hookrightarrow(p^k,r^k)}} = \left(\mathcal{N}_2 || \mathcal{N}_1\right)\Big|_{\substack{(r_2^1,r_1^1)\hookrightarrow(p^1,r^1)\\ \cdots \\ (r_2^k,r_1^k)\hookrightarrow(p^k,r^k)}}$$

V. Construction and illustration

The definitions presented above will help the formal construction of behavioral models in TPN. This construction will serve as a basis of the transformation process within SExPIsTools framework (Figure 1). To better understand the concepts involved in this construction, we must specify the major categories of concepts in RTEPML [2]. At the moment, three of them were selected from a behavioral point of view: concurrent resources (i.e., tasks, interruptions, alarms, etc.), interaction resources (i.e., semaphores, message queues, shared data, events, etc.) and routines (i.e., application code). For the sake of clarity, the construction has deliberately been splitted into four composition operations. The overall construction is a sequence of four operations.

A construction example in TPN is provided to illustrate the method. Figure 2 presents some TPN fragments instantiated with roles (in boxes), prepared for construction. Every operation details the fragments involved in the composition. The mergeable places are represented in double circle and those ready to be merged are connected by a hook-dotted arc with the number of the construction. The roles are assigned to the right above of places. The whole model is describing a monoprocessor application $Proc$ with two periodic tasks $Task_1$ and $Task_2$ sharing the same semaphore Sem_1.

a) Construction for each routine: The routines serve as executive body of concurrent resources. They consist of an ordered sequence of call services. The list of services considered in RTEPML is not exhaustive at the moment. The instructions described in TPN are: activation and termination of task, acquisition and release of semaphore and waiting, notification and inhibition of event.

Let n be the number of call services described following: $\{\mathcal{N}_{S1}, \mathcal{N}_{S2}, \ldots, \mathcal{N}_{Sn}\}$ such that $\forall i \in [1,n], \mathcal{N}_{S_i} = Ins(\mathcal{N}_S, S_i)$ with $\mathcal{N}_S$ the TPN describing a service. The routine construction then implies $n-1$ compositions, each one having m_j mergings of places with $j \in [1, n-1]$. The construction of a routine instance $\mathcal{N}_R$ is given by (1).

Illustration 1 (See Figure 2): In accordance with $\mathcal{N}_R$, $\forall l \in [1,2]$, $\mathcal{N}_{Task_l Body}$ is built from TPNs $\{\mathcal{N}_{Get_l(Sem_1)}, \mathcal{N}_{Release_l(Sem_1)}, \mathcal{N}_{Terminate_l(Task_l)}\}$. This sequence describes in the order, an acquisition of Sem_1, a release of Sem_1 and a termination of $Task_l$.

b) Construction for each entry point of a concurrent resource: Each resource points to a routine described by $\mathcal{N}_R$ previously formed. Only one operation composes $\mathcal{N}_R$ with $\mathcal{N}_{C\lambda} = Ins(\mathcal{N}_C, C_\lambda)$. $\mathcal{N}_C$ is the TPN describing a concurrent resource. The construction of a concurrent resource instance with its executive body $\mathcal{N}_{CR}$ is given by (2) for m mergings.

Illustration 2 (See Figure 2): In accordance with $\mathcal{N}_{CR}$, $\forall l \in [1,2]$, $\mathcal{N}_{Task_l\_withBody}$ is built composing $\mathcal{N}_{Task_l}$ with its entry point $\mathcal{N}_{Task_l Body}$.

c) Construction for concurrent resources: At this stage, concurrent resources must be attached together with the aim of being scheduled by the same processor.

Let q_C be the number of concurrent resources with their composed executive bodies such that $\forall i_C \in [1, q_C]$, each resource is described by $\mathcal{N}_{CR_{i_C}}$ in accordance with $\mathcal{N}_{CR}$ previously formed. The construction then implies $q_C - 1$ compositions, each one having m_{j_C} mergings with $j_C \in [1, q_C - 1]$. The construction of $\mathcal{N}_W$ is given by (3).

Illustration 3 (See Figure 2): In accordance with $\mathcal{N}_W$, $\mathcal{N}_{withoutProc}$ is firstly composed of $\mathcal{N}_{Task_1\_withBody}$ and $\mathcal{N}_{Task_2\_withBody}$.

d) Global construction with processor and interaction resources: Note that the processor is also a shared resource. It will therefore be considered as an interaction resource.

Let q_I be the number of interaction resources considered such that $\forall i_I \in [1, q_I]$, each resource is described by $\mathcal{N}_{I_{i_I}} = Ins(\mathcal{N}_I, I_{i_I})$ with $\mathcal{N}_I$ the TPN describing an interaction resource. Each interaction resource is composed with $\mathcal{N}_W$ previously formed. The global construction then implies q_I compositions, each one having m_{j_I} mergings with $j_I \in [1, q_I]$. The global composition $\mathcal{N}_G$ is given by (4).

Illustration 4 (See Figure 2): In accordance with $\mathcal{N}_G$, $\mathcal{N}_{DeployedApplication}$ is finalized by composing $\mathcal{N}_{withoutProc}$, $\mathcal{N}_{Sem_1}$ and $\mathcal{N}_{Proc}$.

VI. Benefits and Limits

The use of TPNs with roles and the composition based on roles has allowed to detect several errors within the SExPIsTools transformation process. Those errors were bad transformation rules between concepts and roles, bad descriptions of the behavioral fragments.

That has also clarified the chaining of the transformation rules. As a result, a part of the transformation prototype has been rewritten. This approach has increased the confidence in SExPIsTools framework and its generated formal models.

Although SExPIsTools can consider several RTOSs, we have only defined fragments for OSEK/VDX [9] in TPN. Moreover, some complex real-time mechanisms, such as priority ceiling protocol or special queues of message box show the limits of the expressiveness of TPNs. For this reason, we could not model those mechanisms.

VII. Conclusion

An approach has been presented to build a formal model of RTESs taking into account a RTOS description. A new definition has extended the modeling in TPN to compose fragments with roles. The formalized composition will be used as a basis of the transformation process. The process implementation in SExPIsTools is in progress. The framework integrates a modeling language called RTEPML designed to describe the behavior of RTOSs. During the process running, only the description of the target execution platform is considered.

The main idea of this process is to maintain a genericity of implementation. Composition rules introduced in this paper are independant of any RTOSs thanks to role notion. This notion is an essential point of our strategy and brings an advantage in relation to other existing approaches. Future prospects are scheduled in order to take into account other RTOS descriptions. Another important point is the consideration of more

 ISBN: 978-1-61208-307-0

$$\mathcal{N}_R = \Bigg(\Big(\big(\mathcal{N}_{S_1}||\mathcal{N}_{S_2}\big)\Big|_{\substack{(end\_S_1,start\_S_2)\hookrightarrow(S_{S_1\to S_2},\perp)\\(r^2_{S_1},r^2_{S_2})\hookrightarrow(p^2_{S_2},r^2_{S_2})\\\dots\\(r^{m_1}_{S_1},r^{m_1}_{S_2})\hookrightarrow(p^{m_1}_{S_2},r^{m_1}_{S_2})}} ||\mathcal{N}_{S_3}\Big)\Big|_{\substack{(end\_S_2,start\_S_3)\hookrightarrow(S_{S_1S_2\to S_3},\perp)\\(r^2_{S_1S_2},r^2_{S_3})\hookrightarrow(p^2_{S_3},r^2_{S_3})\\\dots\\(r^{m_2}_{S_1S_2},r^{m_2}_{S_3})\hookrightarrow(p^{m_2}_{S_3},r^{m_2}_{S_3})}} \dots||\mathcal{N}_{S_n}\Bigg)\Big|_{\substack{(end\_S_{n-1},start\_S_n)\hookrightarrow(S_{S_1S_2\dots S_{n-1}\to S_n},\perp)\\(r^2_{S_1S_2\dots S_{n-1}},r^2_{S_n})\hookrightarrow(p^2_{S_n},r^2_{S_n})\\\dots\\(r^{m_{n-1}}_{S_1S_2\dots S_{n-1}},r^{m_{n-1}}_{S_n})\hookrightarrow(p^{m_{n-1}}_{S_n},r^{m_{n-1}}_{S_n})}} \quad (1)$$

with $\forall k \in [1,m_j]$ and $n \geq 2$ if $k \geq 2$ then $r^k_{S_1\dots S_j} = r^k_{S_{j+1}}$

$$\mathcal{N}_{CR} = \big(\mathcal{N}_{C_\lambda}||\mathcal{N}_R\big)\Big|_{\substack{(start\_C_\lambda,start\_S_1)\hookrightarrow(S,\perp)\\(end\_C_\lambda,end\_S_n)\hookrightarrow(E,\perp)\\(r^3_{C_\lambda},r^3_R)\hookrightarrow(p^3_R,r^3_R)\\\dots\\(r^m_{C_\lambda},r^m_R)\hookrightarrow(p^m_R,r^m_R)}} \quad (2)$$

with $\forall k \in [1,m]$ if $k \geq 3$ then $r^k_{C_\lambda} = r^k_R$

$$\mathcal{N}_W = \Bigg(\Big(\mathcal{N}_{CR_1}||\mathcal{N}_{CR_2}\Big)\Big|_{\substack{(processor\_CR_1,processor\_CR_2)\hookrightarrow(P_{CR_1\to CR_2},processor\_Proc)\\(r^2_{CR_1},r^2_{CR_2})\hookrightarrow(p^2_{CR_2},r^2_{CR_2})\\\dots\\(r^{m_1}_{CR_1},r^{m_1}_{CR_2})\hookrightarrow(p^{m_1}_{CR_2},r^{m_1}_{CR_2})}} \dots||\mathcal{N}_{CR_{q_C}}\Bigg)\Big|_{\substack{(processor\_CR_{q_C-1},processor\_CR_{q_C})\hookrightarrow(P_{CR_1\dots CR_{q_C-1}\to CR_{q_C}},processor\_PROC)\\(r^2_{CR_1\dots CR_{q_C-1}},r^2_{CR_{q_C}})\hookrightarrow(p^2_{CR_{q_C}},r^2_{CR_{q_C}})\\\dots\\(r^{m_{q_C-1}}_{CR_1\dots CR_{q_C-1}},r^{m_{q_C-1}}_{CR_{q_C}})\hookrightarrow(p^{m_{q_C-1}}_{CR_{q_C}},r^{m_{q_C-1}}_{CR_{q_C}})}} \quad (3)$$

with $\forall k_C \in [1,m_{j_C}]$ and $q_C \geq 2$ if $k_C \geq 2$ then $r^{k_C}_{CR_1\dots CR_{j_C}} = r^{k_C}_{CR_{j_C+1}}$

$$\mathcal{N}_G = \Bigg(\Big(\mathcal{N}_W||\mathcal{N}_{I_1}\Big)\Big|_{\substack{(r^1_P,r^1_{I_1})\hookrightarrow(p^1_{I_1},r^1_{I_1})\\\dots\\(r^{m_1}_P,r^{m_1}_{I_1})\hookrightarrow(p^{m_1}_{I_1},r^{m_1}_{I_1})}} \dots||\mathcal{N}_{I_{q_I}}\Bigg)\Big|_{\substack{(r^1_{PI_1\dots I_{q_I-1}},r^1_{I_{q_I}})\hookrightarrow(p^1_{I_{q_I}},r^1_{I_{q_I}})\\\dots\\(r^{m_{q_I}}_{PI_1\dots I_{q_I-1}},r^{m_{q_I}}_{I_{q_I}})\hookrightarrow(p^{m_{q_I}}_{I_{q_I}},r^{m_{q_I}}_{I_{q_I}})}} \quad (4)$$

with $\forall k_I \in [1,m_{j_I}]$ and $q_I \geq 1$, $r^{k_I}_{PI_{j_I-1}} = r^{k_I}_{I_{j_I}}$

complex RTOSs mechanisms. The use of high-level Petri Nets such as Scheduling TPNs [5] is also planned.

Finally, a more long-term goal is planned to check the correctness of the transformation. A formal comparison between an application model projected on a more abstract platform and a deployed application model generated by SExPIsTools could allow this verification.

REFERENCES

[1] M. Boyer and O.H. Roux, "On the compared expressiveness of arc, place and transition time Petri nets," Fundamenta Informaticae, August. 2008, pp. 88(3):225-249.

[2] M. Brun and J. Delatour, "Contribution on the software execution platform integration during an application deployment process," First Topcased Day, Toulouse, February. 2011.

[3] W. El Hajj Chehade, A. Radermacher, S. Gérard, and F. Terrier, "Detailed Real-Time Software Platform Modeling," Software Engineering Conference (APSEC), 17th Asia Pacific, November. 2010, pp. 108-117.

[4] C. Lelionnais, M. Brun, J. Delatour, O.H. Roux, and C.Seidner, "Formal Behavioral Modeling of Real-Time Operating Systems," ICEIS(2) - Proceedings of the 14th International Conference on Enterprise Information Systems (Special Session on Model Driven Development for Information Systems: Techniques, Tools, and Methodologies - MDDIS 2012), Wroclaw, Poland: June. 2012, pp. 407-414.

[5] D. Lime and O.H. Roux, "Formal verification of real-time systems with preemptive scheduling," Journal of Real-Time Systems, Springer, February. 2009, pp. 41(2):118-151.

[6] M. Merlin, "A study of the recoverability of computing systems," PhD dissertation, Univ. of California, Department of Information ans Computer Science, Irvine, 1974.

[7] J. Miller and J. Mukerji, "Model Driven Architecture (MDA) Guide, version 1.0.1.," Technical report, Object Management Group, June. 2003.

[8] Object Management Group (OMG), "UML Profile for Modeling and Analysis of Real Time and Embbeded systems (MARTE), version 1.1.," Technical report, June. 2011.

[9] OSEK/VDX Group, "OSEK/VDX Operating System Specification, version 2.2.3.," Technical report, February. 2005.

[10] F. Peres, B. Berthomieu, and F. Vernadat, "On the composition of Time Petri Nets," Discrete Event Dynamic Systems, September. 2011, pp. 21(3):395-424.

[11] F. Taïani, M. Paludetto, and J. Delatour, "Composing real-time objects: a case for Petri nets and Girard's linear logic," Object-Oriented Real-Time Distributed Computing, ISORC-2001. Proceedings. Fourth IEEE International Symposium on, May. 2001, pp. 298-305.

[12] F. Thomas, S. Gérard, J. Delatour, and F. Terrier, "Software Real-Time Resource Modeling," Embedded Systems Specification and Design Languages, Lecture Notes in Electrical Engineering, Springer Netherlands, 2008, pp. 10:169-182.

 ISBN: 978-1-61208-307-0

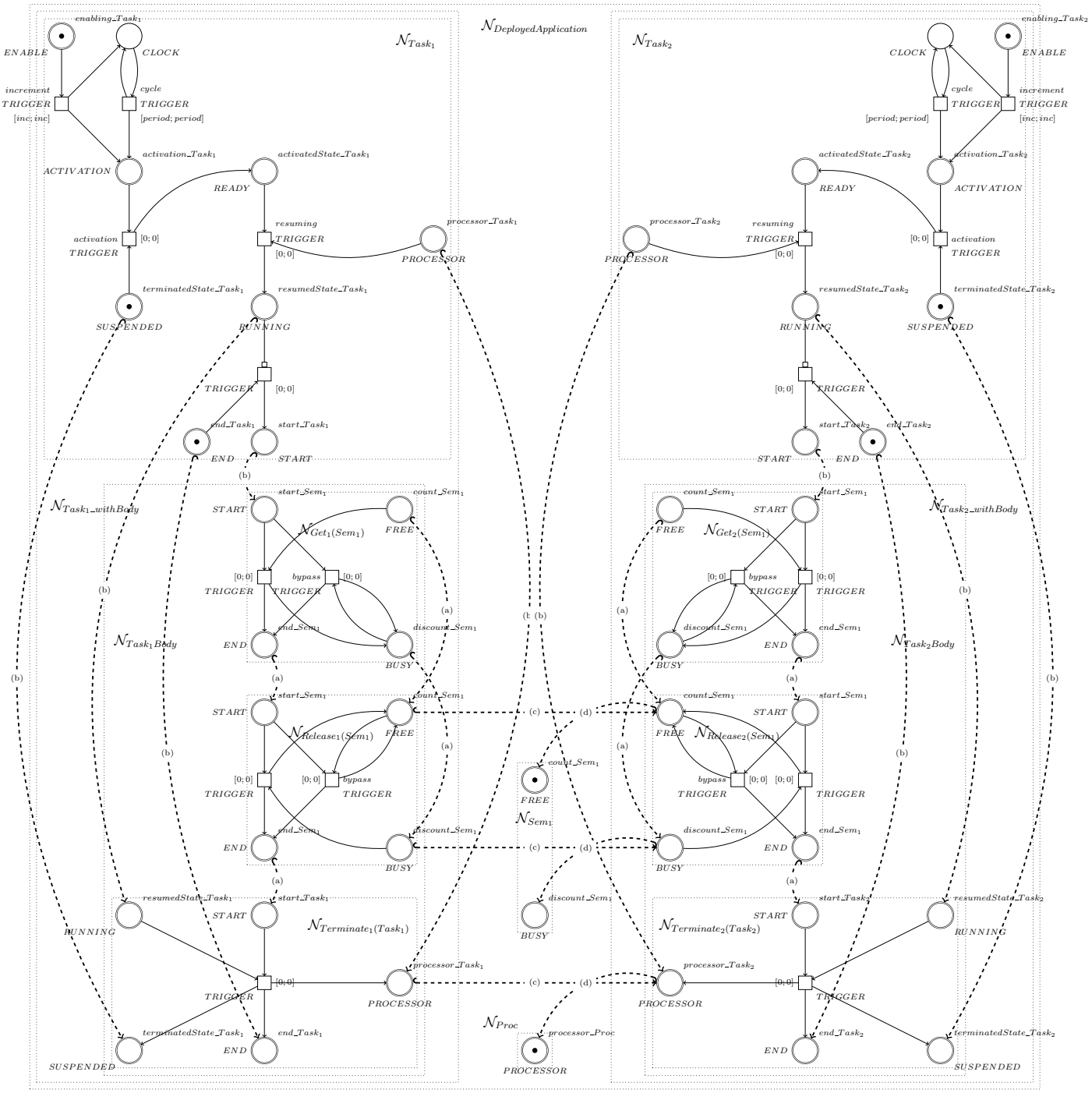

Fig. 2: Deployed application of semaphore sharing composed in TPN