INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 10(2):211-221, April, 2015.

Modelling and Analysis of Mobile Computing Systems: An Extended Petri Nets Formalism

L. Kahloul, A. Chaoui, K. Djouani

Laid Kahloul
LINFI Laboratory, Computer Science Department, University of Biskra
Biskra, 07000, Algeria.
E-mail: kahloul2006@yahoo.fr

Allaoua Chaoui
MISC Laboratory, Computer Science Department, University of Constantine
Constantine, 25000, Algeria.
E-mail: a_cahoui2001@yahoo.fr

Karim Djouani
LISSI Laboratory, Paris Est University, Paris, France
F'SATI at TUT, Pretoria South Africa.
E-mail: djouani@univ-paris12.fr

Abstract: In its basic version, Petri Nets are defined as fixed graphs, where the behaviour of the system is modelled as the marking of the graph which changes over time. This constraint makes the Petri Nets a poor tool to deal with reconfigurable systems as mobile computing systems, where the structure of the system can change as its behaviour, during time. Many extended Petri nets were proposed to deal with this weakness. The aim of this work is to present a new extension of Petri Nets, where the structure of the graph can be highly flexible. This flexibility gives a rich model with complex behaviours, not allowed in previous extensions. The second aim is to prove that even these behaviours are so complex; they can be translated into other low level models (as Coloured Petri Nets [21]) and so be analysed. This translation exploits Dynamic Petri Nets [11] as an intermediary representation between our model and Coloured Petri Nets.
Keywords: Petri Nets , Coloured Petri Nets, Dynamic Petri Nets, Mobile Computing System.

1 Introduction

The development of computer science technologies and increasing user requirements are the major drivers of the birth of sophisticated solutions. Mobility with its soft (code mobility) and hard (device mobility) aspects is one of these solutions. When some disaster menaces a critical system during its execution, it seems a good idea to transfer this system and to save its state to another, more secure site, where it can continue its execution. By soft mobility, we mean a system where code can migrate from one site to another site. Many reasons can cause such migration and many methods and techniques can be used. On the other hand, travelling users who request some computing services need also some specific mobile devices. In this last case, we talk about hard mobility. Applications using code mobility are increasing. Code mobility touches critical domains (military, spacial, medicine). Such domains require that used applications insure a set of properties. Safety, liveness, no deadlock, fault tolerance, and security are example of such required properties. Using formal methods, one can develop systems and proof (or verify) presence or absence for specific properties, in these systems. Formal methods are languages, tools, and approaches allowing specification and verification of systems. Formal languages are based on a well-defined syntax and a formal semantics. Their formal semantics allow developers to

verify specification written in such languages. For some languages, automatic tools are proposed to verify the specifications. Using formal methods in code mobility is not recent. Most currently methods can be considered as derived from processes-algebra [6] or state-transition systems.

As a state-transition model, Petri Nets [18] was proposed to model concurrent and parallel systems. This formalism has a graphic representation and a formal background. Using places, transitions and connecting arcs, this formalism can specify states, actions and transitions between states through which a system evolves. Using Petri nets, one can analyse behavioural or structural properties of a system. To model mobility with Petri nets, the most important contribution can be found in high level Petri Nets. Many extensions have been proposed to adapt Petri nets to mobile systems: Mobile Nets and Dynamic Petri nets [11], Nested Petri Nets [13], Hyper-Petri-Nets [14], Mobile Synchronous Petri Net [15]...

The first idea that has motivated our work was mobile code systems. In these systems, type of resources and their bindings play a central role in the migration procedure. Resources decide also the success or failure of the process. Proposed formal methods founded in the literature do not deal with these aspects and their problems. In our first work, we have proposed a *naive version* of "Labelled Reconfigurable Nets" [23] extended to "Coloured Reconfigurable Nets" in [24]. Our objective was to propose a graphical tool to model mobile code systems in an easy and intuitive way. In these works, we were interested to provide formalisms that model mobility, explicitly (The mobility is modelled through the reconfiguration of the net's structure when some transitions are fired). When trying to offer this quality in a model, we have to deal with the problem of interpreting this reconfiguration formally. In [23, 24], we have introduced specific labelled transitions: "reconfigure-transitions", which reconfigure the structure of the net when they are fired. The first drawback of this solution is that we must provide a specific treatment of these transitions when the model is analysed. In [25], we have proposed an interpretation of Reconfigurable Labelled Nets into a high order Maude (Reconfigurable Maude). The idea was to extend Maude [21] with some "reconfigure rewriting rules". These rules can represent the reconfigure-transitions. Reconfigurable Maude can be used to simulate Reconfigurable Labelled Nets. In the current work, we will present another version that we call Extended "Labelled Reconfigurable Nets", where the label of a reconfigure-transition is defined as a tuple. This tuple contains a set of values which belong to different types. Three specific types are defined: P (for places), T (for transitions), and A (for arcs). These types will contain signed (negative as well as positive) objects (places, transitions, and arcs). The presence of a positive object (resp. negative object) in a label of a reconfigure-transition can be the cause to add (resp. delete) this object to (resp. from) the structure of this net. Using these labels, the structure of the net can be expanded, reduced, or destroyed. Our second *contribution* is the proposition of a method to analyse this model, using Dynamic Nets [11]. Dynamic Nets can be translated into CPN [21], and can be analysed using CPN-tool [26]. We propose the encoding of our model into Dynamic nets. This encoding will be used to analyse this model. This encoding *has been proved*. The encoding and its proof are not presented in this paper.

This paper is organized as follows: The section two presents the formal definition of Extended Labelled Reconfigurable Nets (ELRN), its semantics, and an example of modelling. Section three discusses the analysis issue that we have developed for the analysis of ELRN using a translation of the model into Dynamic Nets [11]. Section four presents a comparison between our work and other similar works, and finally, section five will conclude this paper.

2 Extended labelled reconfigurable nets

Extended Labelled Reconfigurable Nets (ELRN) are an extension of Coloured Petri Nets [21]. In ELRN, the set of transitions is divided into two subsets: *ordinary transitions* (OT) and

reconfigure-transitions (RT). A reconfigure-transition has a label (a tuple of values). The firing of an ordinary transition will change the marking of the Net in an ordinary manner, as in CPN [21]. The firing of a reconfigure-transition changes the marking of the net as well as the structure of the Net. A reconfigure-transition changes the structure of the net, by adding or deleting a node (places, transitions, arcs). To add a place, a reconfigure-transition must have a label which contains: (the name of the place, its initial marking, its input (resp. output) transitions with their incoming (resp. outgoing) expressions). To add a transition, a reconfigure-transition must have a label which contains: (the name of the transition, its guard, its input (resp. output) places with their incoming (resp. outgoing) expressions). Finally, to add an arc, a reconfigure-transition must have a label which contains: (the name of the arc, its labelling expression, and its input/output nodes). Names of nodes (places, transitions, arcs) can be preceded by a negative sign. The presence of negative node in a label causes its elimination from the net (iff it existed in the original net), once the reconfigure-transition is fired. In the following subsection, we present the formal definition of this model, its semantics, and a modelling example.

2.1 Formal definition

Let $Name$ be a set of names. Let P, T, and A be three finite and disjoint subsets of the set $Name$.

An Extended Labelled Reconfigurable Nets $N_{P,T,A}$ is a 10-tuple (Σ, $P\prime$, $T\prime$, $A\prime$, C, G, E, I, L), where:

- Σ: a set of types (Colours). We denote by Σ^* the set of all multi-sets of the set Σ;
- $P\prime$: a set of places; $P\prime \subseteq P$.
- $T\prime$: a set of transitions; $P\prime \subseteq P$. $T\prime = OT \cup RT$ (OT for ordinary transitions, and RT for reconfigure-transitions).
- $A\prime$: a set of arcs. $A\prime \subseteq (T\prime \times P\prime) \cup (P\prime \times T\prime)$. For a place p in $P\prime$ and a transition t in $T\prime$, we can have an arc a in $A\prime$ written $a = (p, t)$ (resp. (t, p)), if it connects p to t (resp. if it connects t to p). We write $(p, .)$ (rep $(t, .)$) to denote the set of arcs that start from p (resp. to denote the set of arcs that start from t).
- C: a colour function associated with each place. $C : P\prime \to \Sigma$. For each place p, C associates a unique colour (type) $C(p)$;
- G: a guard function associated with each transition. $G : T\prime \to Exp$. Where Exp is the set of all Boolean expressions that can be constructed using constants and variables defined in types Σ;
- E: an expression function that associates to each arc a in $A\prime$ an expression $E(a)$. The expression $E(a)$ is a multi-set of $C(p)$ where $a \in (p, .)$.
- I: is an initial state of the net. I$=< M_0, S_0 >$, where M_0 is the initial marking of places P'. $M_0 : P\prime \to \Sigma^*$. S_0 is the initial structure of the net. We take $S_0 = P\prime \cup T\prime \cup A\prime$.
- L: a labelling function which associates to each transition rt in RT a label. We denote by $(P, Exp)^*$ (resp. $(T, Exp)^*$) the set of couples composed of a place in P and an expression (resp. composed of a transition in T and an expression). We denote by P^- the set of names defined in P preceded by a negative sign (idem for T^- and A^-). We denote by ϵ the empty untyped element. A label is a tuple of values. Three kinds of labels can be used:(i) ($P \cup P^-$, $\Sigma^* \cup \{\epsilon\}$, $(T, Exp)^* \cup \{\epsilon\}$, $(T, Exp)^* \cup \{\epsilon\}$) labels a transition which adds a place

to the net, (ii) ($T \cup T^-$, $Exp \cup \{\epsilon\}$, $(P, Exp)^* \cup \{\epsilon\}$, $(P, Exp)^* \cup \{\epsilon\}$) labels a transition which adds a place to the net, and (iii) ($A \cup A^-$, $\Sigma^* \cup \{\epsilon\}$, $Name \cup \{\epsilon\}$, $Name \cup \{\epsilon\}$) labels a transition which adds an arc to the net. So, we have: $L : RT \rightarrow (P \cup P^- \times \Sigma^* \cup \{\epsilon\} \times (T, Exp)^* \cup \{\epsilon\} \times (T, Exp)^* \cup \{\epsilon\}) \bigcup (T \cup T^- \times Exp \cup \{\epsilon\} \times (P, Exp)^* \cup \{\epsilon\} \times (P, Exp)^* \cup \{\epsilon\}) \bigcup (A \cup A^- \times \Sigma^* \cup \{\epsilon\} \times Name \cup \{\epsilon\} \times Name \cup \{\epsilon\})$.

2.2 Semantics

Let N be an Extended Labelled Reconfigurable Nets, and t a transition in T. As in CPN (Coloured Petri Nets) [21], we denote by $^\circ t$ the set of input places for the transition t, and by t° the set of output places for the transition t. Let $I_0 =< M_0, S_0 >$ be the current state of N. Firing t changes I_0 towards $I_1 =< M_1, S_1 >$. We denote this as: $< M_0, S_0 > \rightarrow^t < M_1, S_1 >$. In case of t in OT, we have: $S_1 = S_0$.

Preconditions to fire t

A binding β is a function that assigns some values to some variables. We denote by $E(p,t)[\beta]$ a binding in which every variable in E is assigned to some values depending on β. Now, the transition t can be fired iff there is a binding β on the variables of $E(p,t)$ such that $M_0(p) \leq E(p,t)[\beta]$, for each $p \in^\circ t$, and $G(t)$ is true.

Post-conditions of firing t

After the firing of t, N will transit from its current state I_0 to another state $I_1 =< M_1, S_1 >$. For each p in $^\circ t$, we will have: $M_1(p) = M_0(p) - E(p,t)[\beta]$. For each $p \in t\circ$, we will have: $M_1(p) = M_0(p) + E(t,p)[\beta]$.
If $t \in OT$ then $S_1 = S_0$. If $t \in RT$ then: S_0 (which is $P\prime_0 \cup T\prime_0 \cup A\prime_0$) will be updated to $S_1 = P\prime_1 \cup T\prime_1 \cup A\prime_1$. Three cases are possible:

- t changes $P\prime$:
  - by adding a place p: the label of t must be: (p, m_p, $\{(in\_t_1, in\_e_1), ..., (in\_t_n, in\_e_n)\}$, $\{(out\_t_1, out\_e_1), ..., (out\_t_l, out\_e_l)\}$); where: m_p is the initial marking of p, and (in_t_i, in_e_i) (resp. (out_t_j, out_e_j)) an input (resp. an output) transition with its incoming (resp. outgoing) expressions. So, $P\prime_1 = P\prime_0 \cup \{p\}$, $T\prime_1 = T\prime_0$, and $A\prime_1 = A\prime_0$.
  - or deleting the place p: the label must be ($-p$, ϵ, ϵ, ϵ). So, $P\prime_1 = P\prime_0 \setminus \{p\}$, $T\prime_1 = T\prime_0$, and $A\prime_1 = A\prime_0$.
- t changes $T\prime$:
  - by adding the transition at: the label of t must be: (at, g_{at}, $\{(in\_p_1, in\_e_1), ..., (in\_p_n, in\_e_n)\}$, $\{(out\_p_1, out\_e_1), ..., (out\_p_l, out\_e_l)\}$); where: g_t is a guard, and (in_p_i, in_e_i) (resp. (out_p_j, out_e_j)) an input (resp. an output) places with its incoming (resp. outgoing) expressions. So, $P\prime_1 = P\prime_0$, $T\prime_1 = T\prime_0 \cup \{at\}$, and $A\prime_1 = A\prime_0$.
  - or deleting the transition dt: the label of t must be: ($-dt$, ϵ, ϵ, ϵ). So, $P\prime_1 = P\prime_0$, $T\prime_1 = T\prime_0 \setminus \{dt\}$, and $A\prime_1 = A\prime_0$.
- or t changes $A\prime$:

- by adding the arc $a = (p, t\prime)$: the label must be $(a, e_a, p, t\prime)$; where: e_a is a labelling expression, p the name of a place, and $t\prime$ the name of a transition. So, $P\prime_1 = P\prime_0$, $T\prime_1 = T\prime_0$, and $A\prime_1 = A\prime_0 \cup \{a\}$.
- by adding the arc $a = (t\prime, p)$: $(a, e_a, t\prime, p)$; where: e_a is a labelling expression, p the name of a place, and $t\prime$ the name of a transition. So, $P\prime_1 = P\prime_0$, $T\prime_1 = T\prime_0$, and $A\prime_1 = A\prime_0 \cup \{a\}$.
- or deleting the arc a: the label of t must be $(-a, \epsilon, \epsilon, \epsilon)$. So, $P\prime_1 = P\prime_0$, $T\prime_1 = T\prime_0$, and $A\prime_1 = A\prime_0 \setminus \{a\}$.

2.3 A modelling example

In the example of Fig 1, we make an explicit subdivision of the net into a set of sub-blocs. Each sub-bloc can represent an agent or a site where many agents reside. Each sub-bloc has a title presented on the top of this sub-bloc. We use a specific graphical representation for the reconfigure-transitions, to distinguish them from the ordinary transitions. In Fig 1, we have three agents. $Agent_1$ and $Agent_3$ are immobile agents which existed on two different sites (S_1 and S_2). $Agent_2$ is a mobile Agent, which is located initially on the site S_1. On the site S_1, $Agent_2$ communicates with $Agent_1$ through the communication place C_1. $Agent_2$ receives an information (of a some type that we denote: **Information**) from C_1, then it moves towards the site S_2, where $Agent_3$ is located. On the site S_2, $Agent_2$ passes the information received from $Agent_1$ to the $Agent_3$, through the place C_2. To do the transfer of this information, a new arc (t_{22}, C_2) must be added to the model, and the arc (C_1, t_{21}) must be deleted from the model. To do this reconfiguration in the model, we use the two reconfigure-transitions: rt_1 and rt_2, with two labels: $L_1 =< -(C_1, t_{21}) >$ and $L_2 =< (t_{22}, C_2), t_{22}, C_2, X >$. X is the variable which transfers the information to $Agent_3$. Fig 2 shows the system after the movement of the $Agent_2$, where the system is reconfigured.

The initial marking of the places is $\{M_0(P_{11}) =< inf >, M_0(P_{21}) =< \bullet >\}$, where inf is a data of the type **Information** and $< \bullet >$ represents the constant black-token of the type **Black-token** (A type which contains only one value which is $< \bullet >$). The variable X is also of type **Information**. The non-labelled arcs are implicitly labelled $< \bullet >$.

3 On the analysis of ELRN

Our aim is to offer a way to analyse ELRN models. We have proposed that the analysis of ELRN can be done through the analysis of some equivalent models in CPN (Coloured Petri Nets [21]) or PN (Petri Nets [18]). Petri Nets and Coloured Petri Nets have been studied for many years and have many automatic verification-tools [22]. To profit from these tools, we must show that there is some correct transformation (an *Unfolding*) from the ELRN formalism (as high level nets) towards CPN (and PN). The transformation of ELRN directly into CPN or PN is a hard task; so we propose to prove that the Extended Labelled Reconfigurable Nets (ELRN) models can be encoded into Dynamic Nets (DN) [11]. We use the DN as an intermediary pass between ELRN and CPN.

At this stage, the obtained specification can be transformed into CPN (Coloured Petri Net Model) model and so, be analysed using CPN-tool for example. The unfolding of ELRN models into DN is developed and has been proved.

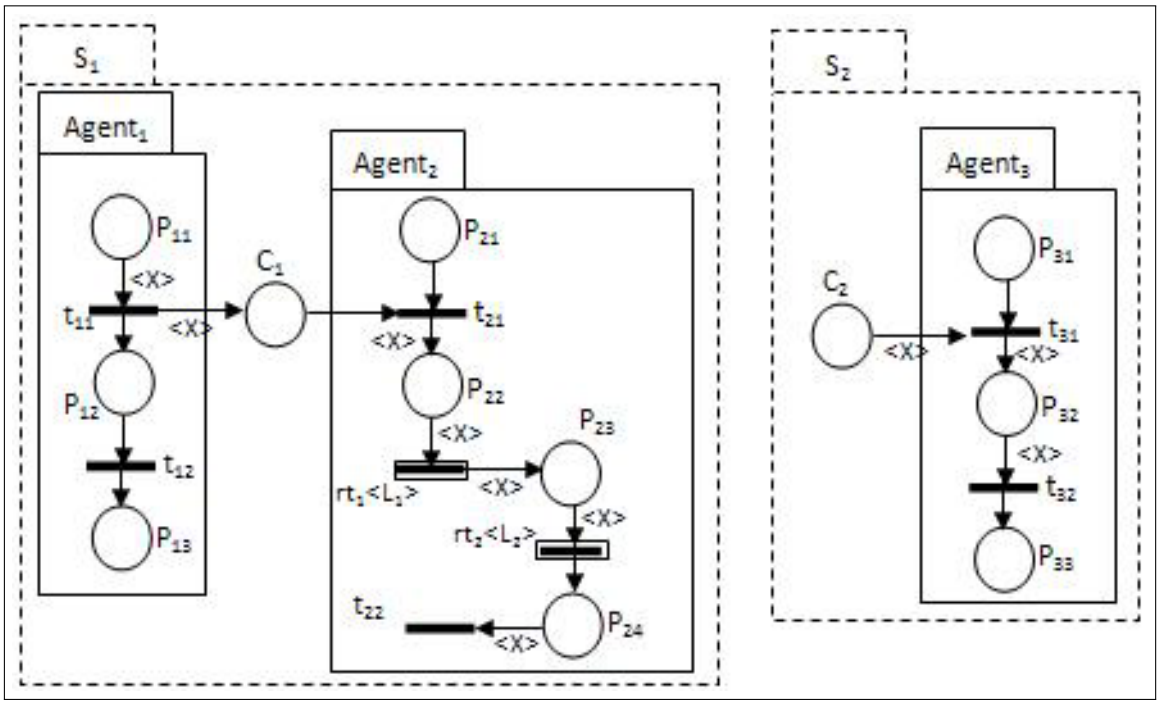

Figure 1: Example of an ELRN.

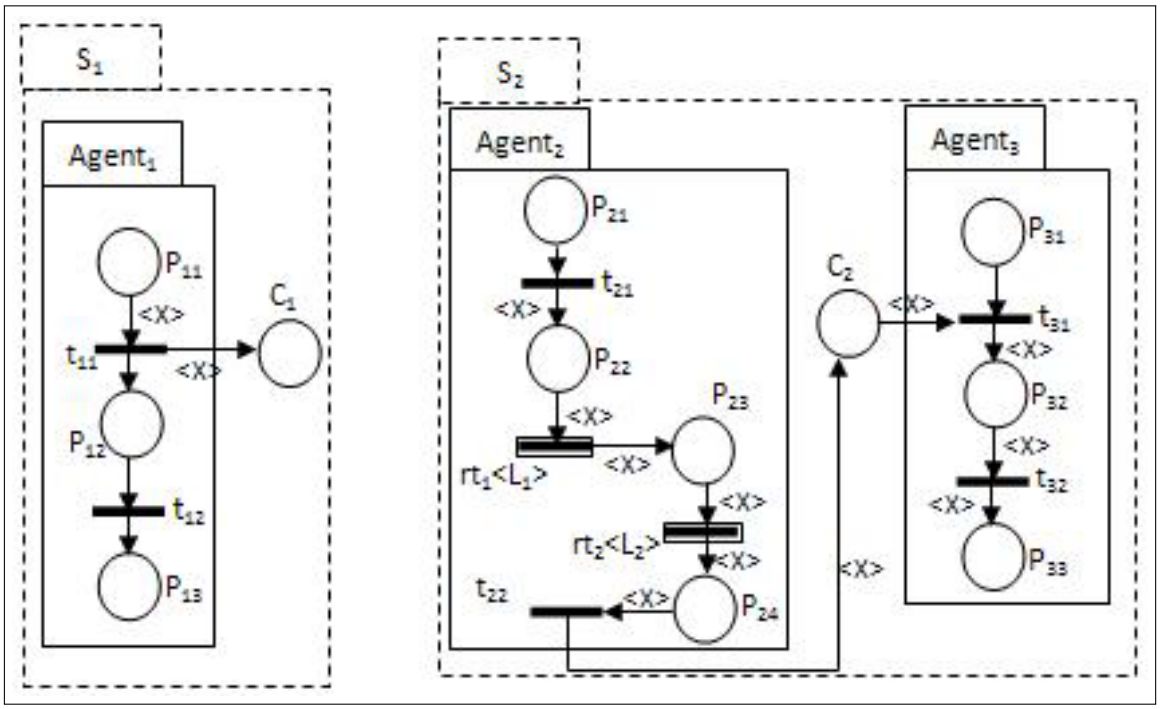

Figure 2: Example of an ELRN (after reconfiguration).

4 Related works

Research on extending Petri Nets (to model systems, with dynamic structure) has provided some remarkable results. We can distinguish between extensions that model mobility in an implicit way (no modification in the structure of the net), or in an explicit way (the reconfiguration, of the net structure, models mobility). The most important propositions are dedicated to mobile systems and mobile agents. In PrN (Predicate/Transition nets) [16], mobile agents are modelled through tokens. These agents are transferred by transition firing from an environment to another. In this work, the structure of the net does not change. The agents are represented as token, so this abstraction does not allow representing some complex behaviour of this kind of agents. In [15], authors proposed MSPN (Mobile synchronous Petri net) as formalism to model mobile systems and security aspects. They have introduced the notions of nets (an entity) and disjoint locations to explicit mobility. A system is composed of set of localities that can contain nets. To explicit mobility, specific transitions are introduced. Two kinds of specific transitions were proposed: new and go. Firing a go transition moves the net from its locality towards another locality. The destination locality is given through a token in an input place of the go transition. In this work, mobility is not also explicit. Mobility is implicitly modelled by the activation of some nets and the deactivation of other nets, using tokens. Migration of an agent is modelled by the deactivation of the net modelling this agent in a locality and the activation of the net that represents this same agent in the destination locality. So, this is a kind of simulation of mobility. In nested nets [21], tokens can be Petri nets themselves. This model allows some transition when they are fired to create new nets in the output places. Nested nets are hierarchic nets where we have different levels of details. Places can contain nets, and these nets can contain also nets as tokens in their places etcetera. So all nets created when a transition is fired are contained in places. So the created nets are not in the same level with the first net. This formalism is proposed to adaptive work-flow systems. Self Modifying Nets (SMN) [18] is an extension of Petri Nets. In this formalism, edges can be labelled by names of places. If the name p is used as the weight of an arc, then this means that the number of tokens to be moved through this arc is equal to the current marking of the place p. So, in self modifying nets, the weights of arcs are dynamic. These weights depend on the current marking of the net. Even SMN offers more computational power than PN; mobility was not the objective of this extension. Though, this formalism was the basis for other more important formalisms like "Reconfigurable Nets" [10]. In "Reconfigurable Net" [10], the structure of the net is not explicitly changed. No places or transitions are added in runtime. The key difference with coloured Petri nets is that firing transition can change names of output places. Names of places can figure as weight of output arcs. This formalism is proposed to model nets with fixed components but where connectivity can be changed over time.

In Object Petri Nets [21], and Elementary Object Nets (EON) [12], tokens can be Petri nets themselves. In an EON, we distinguish between System Nets and Object Nets. Object Nets play the role of object tokens that can appear in places of a System Net. Here, a two-level system modelling technique is introduced. The System Net which represents the external level and the Object Net which represents the internal level can have some synchronous transitions. In this case, these transitions must be fired simultaneously in the two levels. Object Nets used as tokens in a System Net can also interact. EON formalism was proposed to model some kind of systems like: work-flow, flexible manufacturing, and mobile agents. Based on the EON formalism, other proposals were done: Nested Nets [13], Petri Hypernets [14], Nets within Nets [19], etc In Nested Nets [21], firing some transitions creates new nets (called token nets) in their output places. Nested nets are also hierarchic nets, where we have different levels of details. Places can contain nets, and these nets can contain also nets as tokens in their places. This formalism was proposed to adaptive work-flow systems. Adaptive means an ability to modify processes in a structured

way, for example by replacing a sub-process or extending it. Petri Hypernets [14] are proposed to model mobile agents. Mobile agents are modelled as nets. These mobile agents are manipulated by other agents (modelled as nets) who can be also mobile. We call Open Net a net used to model a mobile agent. This open net plays the role of a token in another net; this last one is called hyper-marking Net. As a difference with Valk's proposal [12, 20], the inter-level synchronization in Hyper-Nets is achieved solely by means of exchanging messages. In [17], PEPA nets are proposed, where mobile code is modelled by expressions of the stochastic process algebra PEPA which play the role of tokens in (stochastic) Petri nets. The Petri net of a PEPA net models the architecture of the net, which is a static one. Mobile Petri nets (MPN) [11] extend coloured Petri nets to model mobility. MPN is inspired by join-calculus [4]. The output places of transition are dynamic. The input expression of a transition defines the set of its output places.

In all these formalisms, the structure of the net is not changed and mobility is modelled implicitly through the net's dynamic. In these models, an important work is required from the modeller to model mobility implicitly. MPN are extended to Dynamic Petri Net (DPN) [11]. Mobility in DPN is modelled explicitly, by adding subnets when transitions are fired. However, the Dynamic Petri nets formalism implies some constraints: (i) No transition without input places, (ii) Added nets, to the original net, must not modify the input of an existing transition in the original net, (iii) We can not add a connection between two disconnected existing nodes, (iv) and we cannot delete nodes (place, transition or connection).

In this paper, we have proposed an extension for Petri nets that can be used to model mobility (and in general, reconfigurable systems): Extended Labelled Reconfigurable Nets. Extended Labelled Reconfigurable Nets is more flexible and more expressive and does not imply constraints on the dynamic of the structure. We consider that Extended Labelled Reconfigurable Nets can be used by reconfigurable systems developers with more flexibility than other formalisms. This is due to the feature that it models mobility explicitly through mobility of nodes in the Labelled Reconfigurable Net. Developers can encode mobile aspects of their system directly and explicitly in the EARN formalism.

The power of Petri nets resides in its verification methods. When extending Petri nets, we reach some formalism with a high expressiveness, but the analysis becomes more complex or even impossible. Developers of new formalisms must propose analysis techniques. Mostly, they are proposing some translation (or encoding) of their formalisms into some well-known formalism or approach in modelling domain. Such translation allows the analysis of the new formalisms models using techniques of well-known formalisms. The most famous encoding can be found in the unfolding of Petri nets into automaton to apply model-checking, and then the unfolding of CPN [21] (Coloured Petri Nets) into PN [18] (Petri Nets) to analyse some properties that are not analysed on the CPN directly. We can find other works, in literature. In [22], author authors studied equivalence between the join calculus [4] and different kinds of high level nets. They proved the equivalence between Reconfigurable nets (RN) [10] (an extension version of PN) and the join calculus. This equivalence allows to interpret RN into join calculus and to verify those using join-calculus tools. In [19], Petri nets are translated into linear logic programming. This translation can be used to analyse Petri nets using Prolog model-checker. Authors of [20] encoded Synchronous mobile nets (SMN) [15] into rewriting logic [22]. This encoding allows the use of Maude [21] to verify SMNs specifications.

In this paper, we have discussed an encoding of ELRN behaviours into Dynamic nets [11]. This encoding was proved to be correct. The advantage of such encoding resides in the possibility to encode Dynamic nets into CPN (coloured Petri Nets). So, ELRN can be translated into CPN. Once translated into CPN, ELRN nets can be analysed using CPN verification tools.

5 Conclusion

Mobile Systems are systems with a dynamic structure. Their structure changes as they are executed. This class of systems can be found in many domains of our life. Mobile robots used to explore hostile environment, mobile agents used in the internet or in distributed systems, mobile nodes in a mobile wireless networks ... All these systems can be considered as reconfigurable systems. The use of these systems is in expansion for many reasons: their efficiency, their abstractions for the designer, their flexibility ...These characteristics make these systems in the kernel of many critical systems: aeronautics, military, medicine, commerce... The design of these systems becomes a critical activity. Their reliability and their correctness are crucial. To ensure the correctness of these systems, formal methods seem to be an adequate solution. Using formal methods, the designer specifies the system in a formal language. A formal language has a well defined syntax, and formal semantics which allows the verification of properties of the designed system. We found in the literature, many formal methods. Classical formal methods (proposed for classical systems) are well defined and are mature. However, these classical formal methods have not the expressiveness to specify reconfigurable systems. The use of the classical methods makes the designer's task a hard task. Extended versions are proposed to deal with the idea of reconfigurable systems. In the literature, we can find two principal classes: Processes algebra based methods, and state-transition based methods.

State-transition based methods can be found in extensions of Petri nets model. Petri nets are an elegant model for concurrency. With its graphical representation and its formal background, it was used to specify and verify concurrent multi-processes systems. The classical model has not the power of expressiveness to deal with current aspects such as mobility. To take benefits from the power of the model in mobility domains, several works have been proposed. These works try to extend Petri nets with the same ability to specify mobility (and more generally: reconfigurability).
In this paper, we have presented the Extended Labelled Reconfigurable Nets formalism. The formal definition of this formalism, its semantics and a modelling example are presented. The encoding of this formalism into another formalism Dynamic Nets [11] was proved using and offers a method to do the analysis of this model.

As perspectives of the current work, we propose the below axes as open domains:

- The experimentation of ELRN in the modelling of mobile systems: Mobile agents systems, mobile networks, ... This modelling work can prove the power of our formalism and shows its shortcomings and so allow us to introduce necessary adaptations;

- The work on automatic verification: The translation of dynamic nets into coloured Petri nets is presented in [11]. We are working on the development of a tool-kit to implement this translation. The encoding presented in section three is formal and proved to be correct; so it is possible to think of an implementation of this last encoding also;

- In the current time, complexity and decidability issues are not yet studied. These aspects are important, once a new formalism is proposed. These issues will be also developed in our future works.

Bibliography

[1] D. Sangiorgi and D. Walker (2001); *The π-Calculus: A Theory of Mobile Processes*, Cambridge University Press.

[2] F. Cédric, G. Gonthier (2000); The Join Calculus: a Language for Distributed Mobile Programming, *Applied Semantics*, International Summer School, APPSEM 2000, Caminha, Portugal, Sept. 2000, LNCS 2395, 268-332.

[3] J.C.M. Baeten (2003); Over 30 years of process algebra: Past, present and future, in L. Aceto, Z. Ésik, W.J. Fokkink, and A. Ingólfsdóttir, editors, *Process Algebra: Open Problems and Future Directions*, vol. NS-03-3 of BRICS Notes Series, 7-12.

[4] F. Cédric, G. Gonthier, J. J. Lévy, L. Maranget, D. Rémy (1996); A calculus of mobile agents, *Proc. 7th International Conference on Concurrency Theory (CONCUR'96)*, 406-421.

[5] E. Badouel, O. Javier (1998); Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes within Workflow Systems, Rapports de recherche INRIA, ISSN 0249-6399.

[6] A. Asperti, N. Busi (2009); Mobile Petri Nets, Technical Report UBLCS-96-10, Department of Computer Science University of Bologna, *Mathematical Structures in Computer Science journal*, 19 (6): 1265-1278.

[7] R. Valk (1998); Petri Nets as Token Objects: An Introduction to Elementary Object Nets, *Applications and Theory of Petri Nets*, LNCS vol. 1420, 1-25.

[8] I.A. Lomazova (2001); Nested Petri Nets, Multi-level and Recursive Systems, *Fundamenta Informaticae*, 47(3): 283-293.

[9] M. A. Bednarczyk, L. Bernardinello, W. Pawlowski, L. Pomello (2004); Modelling Mobility with Petri Hypernets, *17th Int. Conf. on Recent Trends in Algebraic Development Techniques, WADT'04*, LNCS vol. 3423.

[10] F. Rosa-Velardo, O.M. Alonso, D. F. Escrig (2005); Mobile Synchronizing Petri Nets: a choreographic approach for coordination in Ubiquitous Systems, *1st Int. Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems*, MTCoord'05. ENTCS 150.

[11] Dianxiang Xu, Yi Deng (2000); Modeling Mobile Agent Systems with High Level Petri Nets, *IEEE International Conference on Systems, Man, and Cybernetics*, 5: 3177-3182.

[12] S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo (2003); PEPA nets: a structured performance modelling formalism, *Performance Evaluation*, 54(2):79-104.

[13] C.A. Petri (1962); KommuniKation mit Automaten, Schriften des IIM Nr.2, Institut fr Instrumentelle Mathematik, Bonn (1962). English translation: Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, vol. 1, suppl. 1, 1966.

[14] R. Valk, Self Modifying Nets (1978); A Natural Extension of Petri Nets, *Proceeding of ICALP'78, Lecture Notes in Computer Science*, 62: 464-476.

[15] M. Khler, D. Moldt, H. Rlke (2003); Modelling mobility and mobile agents using nets within nets, In W. van der Aalst and E. Best, eds., *Applications and Theory of Petri Nets 2003, Proceeding*, vol. 2679 of LNCS, 121-139.

[16] R. Valk (2004); Object Petri nets: Using the nets-within-nets paradigm, In Jrg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, eds., *Advances in Petri Nets: Lectures on Concurrency and Petri Nets*, vol. 3098 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, 819-848.

[17] K. Jensen (1994); An Introduction to the Theoretical Aspects of Coloured Petri Nets, In J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.), *A Decade of Concurrency, Lecture Notes in Computer Science*, 803: 230-272.

[18] http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html.

[19] M. Buscemi, V. Sassone (2001); High-Level Petri Nets as Type Theories in the Join Calculus, *Proc. of Foundations of Software Science and Computation Structure (FoSSaCS '01)*, LNCS 2030.

[20] F. Rosa-Velardo (2007); Coding Mobile Synchronizing Petri Nets into Rewriting Logic, *Electronic Notes in Theoretical Computer science*, 174(1): 83-98.

[21] M. Clavel, F. Durn, S. Eker, P. Lincoln, N. Mart-Oliet, J. Meseguer, J. Quesada (1999); Maude: specification and programming in rewriting logic, *SRI International*, http://maude.csl.sri.com.

[22] J. Meseguer (1992); Conditional rewriting logic as a unified model of concurrency, *Theoretical Computer Science*, 96 (1): 73-155.

[23] L. Kahloul, A. Chaoui, Code mobility modeling: a temporal labelled reconfigurable nets, *Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, MOBILWARE 2008*, Innsbruck, Austria, February 13 - 15, 2008. ACM International Conference Proceeding Series 278.

[24] L. Kahloul, A. Chaoui (2008); Coloured Reconfigurable Nets for Code Mobility Modeling, *International Journal of Computers, Communications & Control*, ISSN 1841-9836, Suppl. issue, 3(S): 358-363.

[25] L. Kahloul, A. Chaoui, LRN/R-maude based approach for modeling and simulation of mobile code systems, *Ubiquitous Computing and Communication Journal (UbiCC journal)*, Vol. 3 No. 6, 12/20/2008. http://www.ubicc.org/search_advanced.aspx.

[26] http://wiki.daimi.au.dk/cpntools/cpntools.wiki.