Using High Level Nets for the Design of Reconfigurable Manufacturing Systems

Laid Kahloul[1], Allaoua Chaoui[2], Karim Djouani[3], Samir Bourekkache[1], Okba Kazar[1]

[1] LINFI Laboratory, Computer Science Department, Biskra University, Algeria.
kahloul2006@yahoo.fr
[2] MISC Laboratory, Computer Science Department, Constantine 2 University, Algeria.
a_chaoui2001@yahoo.com
[3] LISSI Laboratory, Paris-Est University, France.
djouani@upec.fr

Abstract. Reconfigurable systems (RSs) are systems were the structure can be changed during the execution of the system. Reconfigurable manufacturing systems (RMSs) represent one of the most prominent successes in the RSs technology. Reconfiguration in RMSs can be motivated by many reasons: a new requirement in the production process, to avoid some problems caused by machines breakdowns, etc. RMSs offer flexibility, productivity and efficiency in plants and production lines. Though, the design, realisation and verification of RMSs seem to be hard tasks and imply innovative approaches. High level Petri Nets supply the ability to design these systems and to analyse their properties. In this paper, we apply Reconfigurable Object Nets (RONs) for the modelling, simulation and analysis of reconfigurable manufacturing systems. We present an experience where the reconfiguration process, in RMSs, is specified explicitly as a "Place/transition nets transformation", the simulation is realised using the RON-editor tool, and the analysis exploits the TINA-tool (TImed Nets Analyser tool).

1 Introduction

Manufacturing Systems (MSs) [1] are widely used in industry. They are characterised by their hybrid aspect and their complexity. A manufacturing system is, usually, composed of a set of components: machines, robots, conveyors, buffers, and eventually humans. These components co-exist and interact to produce some products. Interactions, between these components, are done explicitly through exchanged messages or implicitly through travelling products during the manufacturing process. The success of a manufacturing system is based on the quality of each component and on the quality of the interactions. Reconfigurable manufacturing systems (RMSs) [2, 3] are MSs where the components and their interaction can change over time. Thus, the structure of the system is no more static but its structure is dynamic. This reconfigurability makes the system more

flexible, allows its adaptation for new events, and so that enhances its productivity. In a reconfigurable manufacturing system, the flow changes dynamically and the components are self-reconfigured to answer new requirements or to handle damages. According to [4], the RMSs guarantee three abilities in the production systems: capacity ability, functionality ability, and cost ability. Capacity ability allows the system to adapt the production's quantity to the dynamic requirement of the market. Functionality ability allows the system to change their functionality and so the quality of the product. Cost ability allows RMSs to reduce the cost of the production and the cost of the reconfiguration process.

Although the advantages of reconfigurability in RMSs, it makes the RMSs more complex, and their development becomes a hard task. Reconfigurability imposes new challenges to the developers, where new kinds of errors and anomalies will probably appear. One of the most critical questions, when designing a Reconfigurable Manufacturing System, is about the properties of the system after each reconfiguration process. When the system is reconfigured, the new configuration must still satisfying the well properties satisfied in the former configuration but bad properties must be avoided. In order to guarantee such constraints, sophisticated verification processes are required. Verification of RMSs can be done using classical techniques, used for classical manufacturing systems as Petri Net (PN) [5]. This formalism attracted, early, researchers in manufacturing systems domain [6-9]. However, with its classical definition as a non dynamic formalism (characterised by its rigid structure); this classical formalism is not suitable to catch some important aspects in RMSs as reconfiguration ability. To enhance Petri nets formalism, new extensions were proposed to deal with reconfiguration process and for the study of dynamic systems [10-14].

The global objective, of our work, is to build a formal approach that can be used to specify, simulate, and analyse reconfigurable manufacturing systems. The approach uses the Reconfigurable Object Nets (RONs) formalisms [15] as a formal model, exploits the RON-tool [16] to simulate interactively the system, and uses the TINA-tool (TIme Petri Net Analyzer) [17] to verify the reached configurations. The RONs are high level Petri Nets, where the tokens can be also nets (called token-nets). The token-nets can change their structure due to reconfiguration rules (modelled also as other tokens called "token-rules" in the RON). The RONs formalism has two advantages; firstly it finds its mathematical background in a well-founded theory (graph transformation theory [18]), secondly it has been implemented in many automatic tools which allow the simulation and the verification of the system (RON-tool [16], ReConnect [30]). The current paper presents an experimentation on a case-study inspired from [19].

This paper is organised as follows: section 2 will present related work. Section 3 will detail the Reconfigurable Object Nets formalism and its mathematical background in graph transformation techniques. Section 4 will present the approach proposed to specify RMSs using RONs, and then demonstrates this approach on a case study. Section 5 will treat the simulation and the verification processes, using automatic tools (RON-tool and TINA-tool). Finally, Section 6 will conclude this paper.

2 Related Work

Several works have used PNs and their extensions in the design and verification of RMSs (Reconfigurable Manufacturing Systems). In this section, we examine some recent works which are close to our work. We classify these works into two principal classes: works where PNs (Petri Nets), without dynamic structure, are applied to RMSs ([19-22, 37]), and works which have exploited High level Petri nets (with dynamic structure) for RMSs ([11, 18, 23-25, 28-30, 33- 36]).

The first class of works finds its motivation in the maturity and stability of the used formalisms: p-time PNs in [20], Coloured PNs in [21], and Coloured Timed Object Nets in [19]. In this category of extensions, some researchers treated the reconfiguration in a modular way to facilitate the building of new models after reconfiguration. They enrich PNs with oriented object concepts (derivation, inheritance) or the modularity concept to overcome the reconfigurability complexity. Authors of [19] used coloured timed oriented object nets (CTOONs) to facilitate reconfiguration of the PNs models. In CTOONs, the Petri nets models are seen as objects in classes, and where new objects can be derived from other objects. The authors of [19] consider the reconfiguration process in RMSs as a derivation activity in the CTOONs model. In [22], the authors proposed ITPNs (Intelligent Token Petri Nets). In the ITPNs formalism, tokens are enriched with time and knowledge. Transitions in an ITPNs model can be disabled when a token is consumed in the model. The knowledge enclosed in a consumed token decides which transitions must be disabled. A synthesis process is proposed to construct new nets from other nets. This process facilitates the definition of new models from existing ones. However, no mechanism is included in the ITPNs to realise this reconfiguration. Thus, the dynamic of the structure is not implemented in the net itself. In [37], the authors used coloured timed Petri nets in the modelling of RMSs. In this work, the authors introduced a mechanism to define reconfigurability in the CTPN formalism, yielding to a new formalism supporting reconfiguration. This mechanism involves reconfigurable transitions, inhibitor arcs, and specific places (machines class). However, the reconfigurable mechanism is not the same used in our proposal. We believe that the reconfiguration in RONs is more intuitive and makes the model more expressive and more suitable for the RMSs. The power of these models [19-22, 37] resides in the existence of well-founded analysis techniques, where many properties are decidable. Many automatic tools are proposed to model, simulate and analyse systems using these formalisms. The major lack in these approaches is the absence of the ability to represent explicitly and intuitively reconfiguration of the system. In our work, we are interested to use formalisms where the reconfiguration of the system can be modelled, explicitly, through the dynamic structure of the formalism. Thus, our work can be inscribed in the second category of works.

In the second class of works, the used "Petri Net formalism" is enriched by a mechanism to reconfigure itself, when necessary. Thus, the PNs model is more intuitive and more natural to support the modelling of Reconfigurable Manufacturing Systems (RMSs). In this class of works, we find several variants of extensions proposed for PN's. Each variant proposes a specific mechanism to

provide the reconfiguration of the PN's structure. The most popular variants find their origins in Valk's works [28, 29] where a notion of "object token" is introduced. However this "object token", proposed by Valk, has not the ability to change its structure. The proposed extensions, for Valk's proposal, have tried to introduce reconfigurability in the structure of the object nets through two basic mechanisms: graph transformation (yielding to: Reconfigurable Petri Nets [26], RONs (Reconfigurable Object Nets) [15], Reconfigurable Petri Nets [25]), or rewriting rules (yielding to: Badouel's reconfigurable Petri nets [11], Improved Net Rewriting Systems (INRSs) in [23], Hybrid Reconfigurable Petri Nets (HRPNs) [24]). This reconfiguration expands the application of Petri Nets to several systems where the structure is dynamic. In the following paragraphs, we will highlight some works that we consider similar to our work in their objectives or in their applied techniques.

We consider that the first work where graph transformation, as a reconfigurability mechanism, was applied to PNs can be found in [18]. In this former work, graph grammars have been used to define the PNs transformations rules, and as an example the authors used manufacturing systems. However the aim, of this work, was not the design of"Reconfigurable Manufacturing Systems", but only the refinement of Manufacturing Systems. Indeed, the objective was not to provide an approach for the specification and verification of RMSs; but only a study on the refinement of manufacturing systems, using PNs transformations.

In fact, a more close work to our work is the one proposed by Li et al. [23]. The authors developed a new formalism INRSs (Improved Net Rewriting Systems), which are based on Badouel's reconfigurable Petri nets [11]. In their work, the authors proposed a hybrid approach, which combines UML.2's activity diagrams [32] and INRSs formalism, to design RMSs. We can identify three major differences between this work and our current work. Firstly, they used the INRSs formalism which is based on the idea of "rewriting rules" proposed by Badouel [31]. However, the formalism used in our work is based on "graph transformation theory" applied to PNs [18]. This theory was proposed before "rewriting rules" and has been applied and studied in many works. Two advantages motivate the choice of graph transformation theory; the first one is the existence of an important work in developing software tools supporting these transformations, and the second advantage can be found in the application of graph transformation theory to a variety of PNs kinds: P/T nets, Algebraic High level nets, and Coloured Petri Nets. For all these kinds of PNs, many results about properties conservation, during transformations, have been proved (a set of conserved properties during transformation can be found in [18]). The second difference between our work and the work in [23] is that the RONs (Reconfigurable Object Nets) formalism lets us make a one "connected model" which represents: (i) all the system's configurations set, (ii) the applied transformations rules, and (iii) the dynamic at micro-level (functioning of the configuration) and the dynamic at macro-level (the reconfiguration in the system). All these aspects are represented in one model which is the RON-model. This is not the case in the proposed approach in [23] where an INRSs represents one configuration with its

set of rewriting rule. The third difference, and which is an important motivation in our choice, is the availability of software tools that can be used in the simulation and analysis of the RONs models (RON-tool [16], ReConNet [30]).

On another level, authors of [25, 33] applied PNs transformations techniques but for another purpose then RMSs design. In [25] the authors used Reconfigurable P/T nets to model mobile ad-hoc networks. The system modelled is designed for an archaeological disaster/recovery mission. In this mission, a set of teams cooperate and their behaviour can be updated to new situations. The formalism proposed, in [25], is not the same one proposed in [23]. Reconfigurable P/T nets, based also on graph transformation theory, represent the building blocks of Reconfigurable Object Nets which is exploited in our current work. The authors of [25] have not treated the case of manufacturing systems, and so no approach for this kind of systems was proposed; nevertheless, the dynamic aspect of their system is similar to the one presented here, because the same idea is applied: "graph transformation". In [34], the authors used Petri Nets in the development of "Dynamically Reconfigurable Embedded Systems". They exploit the Petri Nets to provide a first specification which will be transformed, after, to generate code. This work was interested to embedded systems (i.e. microcontrollers used in WNS) and their reconfigurability rather than manufacturing systems. The formalism used is a class of basic Petri nets: Work flow PN (WFPN [35]). The reconfigurability within the system is formulated as migrating of nets in the "nets-within-nets" formalism [36]. The Nets-within-nets formalism is considered as the basic of Object Petri Nets [36] used after in the proposition of RONs (used in this paper). This work deals with mobility rather than hardware reconfigurability. Finally, the authors of [33] exploited the RONs formalism for the modelling of mobile agents, used in the mobile maintenance of manufacturing system. This last work used the same formalism used in our present work, but it treats also mobility rather than reconfigurability.

In this paper, we propose to use the RONs formalism to specify, simulate and verify the Reconfigurable Manufacturing Systems. Through an example, we present the requirements for this modelling and how the model can be constructed. One of the advantages of the RONs is the availability of dedicated automatic tools (as RON-tool used in our proposal) to simulate and analyse the constructed models.

3 Reconfigurable object nets

3.1 An informal presentation

Reconfigurable Object Nets (RONs) [26] was introduced firstly in [15], as High Level Nets with Nets and Rules as Tokens. In RONs, we distinguish between two levels in the Net (the system level and the token level) and two classes of tokens (token-nets and token-rules). A place in the system level can contain token-nets or token-rules. A token-net is a P/T net which can move from place to place in the system. During its moving, the token-net's marking can change as well as its structure. Transitions in the system level decide about the movement of

token-nets, as well as if the marking or the structure of these token-nets will change. To change the marking of a token-net, the model must have a transition in the system level which will trigger a transition in the token-net level. However, to change the structure of a token-net, a token-rule is required to specify how this structure will be changed. Hence, the token-rule decides how the structure of the token-net will change when some transition, in the system level, is fired. In RONs, reconfiguration of the structure concerns only the token-nets and not the whole net in the system level. This reconfiguration is defined through a set of token-rules, inspired from graph transformation techniques.

Indeed, graph transformation techniques allow the formulation of two basic constructions: **union** and **transformation**, on Place/Transition Nets (P/T nets). Informally, the union construction takes two Nets N_1 and N_2 and yields another net N_3, but the transformation construction takes one P/T net N_1 and yields another net N_2. In RONs formalism, these two constructions are the two basic reconfigurable techniques for P/T nets. Union and transformation are based on the **morphism** concept defined over P/T nets. In the following paragraphs, we will formalise the necessary concepts for our proposal: Place/Transition nets, morphisms over P/T nets, union, transformation, and finally RONs.

3.2 Place/Transition nets (P/T nets)

A place/transition net is a quadruplet $(T, P, Pre, Post)$, where:

- T: is a finite set of transitions;
- P: is a finite set of places;
- Pre (for pre-domain) and $Post$ (for post-domain) are the two mappings defined as: $Pre, Post : T \to P^{\oplus}$

The set $P^{\oplus}$ is the set of finite multi-sets over the set P. An element w in $P^{\oplus}$ can be written as the sum: $w = \Sigma_{p \in P} \lambda_p \times p$, where: λ_p is a natural number ($\lambda_p \in N$). We can also consider w as a function: $w : P \to N$.

3.3 Morphisms over P/T nets

Given two P/T nets: $N_1 = (T_1,\ P_1,\ Pre_1,\ Post_1)$ and $N_2 = (T_2,\ P_2,\ Pre_2,\ Post_2)$. A morphism f between the two nets N_1 and N_2 is a function: $f : N_1 \to N_2$. We have: $f = (f_T, f_P)$, such that: $f_T : T_1 \to T_2$, and $f_P : P_1 \to P_2$ are two morphisms which map transitions into transitions and places into places, respectively. f_T and f_P satisfy:

1. $Pre_2 \circ f_T = f_P^{\oplus} \circ Pre_1$
2. $Post_2 \circ f_T = f_P^{\oplus} \circ Post_1$

The diagram (on Figure 1) [15] summarizes the above concepts.

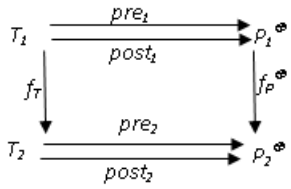

Fig. 1. Morphisms on P/T nets [15]

3.4 Union of P/T nets as a pushout

Based on the morphisms defined over P/T nets, it is possible to define a specific construction which is the pushout (or union) of two P/T nets. Let $N_1 = (T_1, P_1, Pre_1, Post_1)$, $N_2 = (T_2, P_2, Pre_2, Post_2)$, and $I = (T_0, P_0, Pre_0, Post_0)$ be three P/T nets, with the two morphisms: $f : I \to N_1$ and $g : I \to N_2$. The net I is said a common interface between N_1 and N_2. The union of N_1 and N_2 is the Net $N = (T, P, Pre, Post)$, defined using the two morphisms: $f' : N_1 \to N$and $g' : N_2 \to N$. We write: $N = N_1 +_I N_2$. The operator $+_I$ is called the pushout construction (see the Figure 2) or the gluing operator.

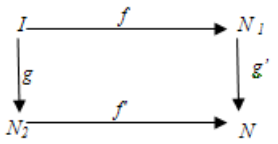

Fig. 2. Union of P/T nets [15]

The net N is constructed as follows:

1. $T = T_1 +_{T_0} T_2$ is the disjoint union of T_1 and T_2, where we glue together all the transitions in: $\{f_T(t), g_T(t)\}_{t \in T_0}$;
2. $P = P_1 +_{P_0} P_2$. P is the disjoint union of P_1 and P_2, where we glue together all the places in: $\{f_P(p), g_P(p)\}_{p \in P_0}$;
3. $Pre(t) = \begin{cases} Pre_1(t_1)\, if\, g'_T(t_1) = t \\ Pre_2(t_2)\, if\, f'_T(t_2) = t \end{cases}$
4. $Post(t) = \begin{cases} Post_1(t_1)\, if\, g'_T(t_1) = t \\ Post_2(t_2)\, if\, f'_T(t_2) = t \end{cases}$

3.5 Transformations of P/T nets as a double pushout rule

Based on the P/T gluing (or pushout) construction, the P/T transformation is constructed as a double pushout. Let L, K, R, and C be four P/T nets. A transformation $f : N_1 \to N_2$ transforms the P/T net N_1 to the P/T net N_2

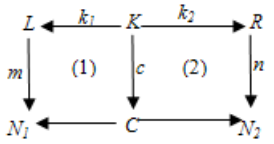

Fig. 3. Double pushout [15]

using the rule $r = (L, K, R)$ and the match $m : L \to N_1$ iff we have the double pushout of the Figure 3.

On Figure 3, k_1, k_2, m, c, and n are morphisms, thus, the P/T net C is called the context of the transformation and it satisfies the following conditions:

1. $T_C = (T_1 \setminus m_T(T_L)) \cup m_T(k_{1_T}(T_K))$;
2. $P_C = (P_1 \setminus m_P(P_L)) \cup m_P(k_{1_P}(P_K))$;
3. $Pre_C = Pre_{1|T_C}$ (The relation Pre_C is the subset of Pre_1 which concerns only the set of transitions: T_C);
4. $Post_C = Post_{1|T_C}$ (The relation $Post_C$ is the subset of $Post_1$ which concerns only the set of transitions: T_C);

3.6 Reconfigurable object nets (RON)

A Reconfigurable Object Nets (RONs) [15] is a high level net where places contain two kinds of tokens: token-nets and token-rules. Token-nets are P/T nets and token-rules are "double pushout" production rules. In the RON model, firing a transition can trigger the movement of a token-net from its current place to another place. Despite this movement, the transition can change the structure of the token-net by applying a token-rule. Indeed, a transition can transform the structure of a token-net as well as it can unify two token-nets.

4 Modelling and simulation of reconfigurable manufacturing systems

4.1 The case study

Let us consider a system inspired (with some modifications) from the one presented in [19]. This system is composed of two manufacturing cells (MC_1, MC_2), a storage AS/AR (Automated Storage and Retrieval System), and an AGV (Automated Guided Vehicle). The system produces a final product A, and uses two raw materials R_1 and R_2 (see the Figure 4). The flow starts in MC_1 and then passes to MC_2.

MC_1 contains a CNC (Computerized Numerical Controlled) *lathe machine*, a CNC *milling machine*, a robot, and a buffer. In MC_1 (see the Figure 5), R_1 and R_2 start in the lathe machine then the results are processed in the milling machine. MC_2 contains an assembly machine (which assembles the two products

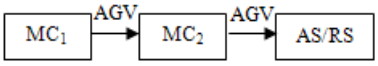

Fig. 4. Manufacturing of product A

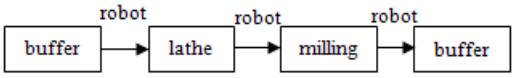

Fig. 5. Flow in MC1

into one product A), a robot, and a buffer. The flow in MC_2 is depicted on Figure 6.

Reconfiguration: during the life of the system, the plan will meet two reconfigurations. Firstly, a new type of product is required: Product B. Product B requires the flow depicted on Figure 7, where the assembly is done before the lathe and the milling.

The second reconfiguration (concerns the production of B) occurs when a new cell MC_3 (inspection cell) is introduced in the system (Figure 8). The inspection cell contains: a Coordinate Measuring Machine (CMM) and a set of buffers.

4.2 The modelling process

The modelling using RONs (Reconfigurable Object Nets) requires the definition of the two levels: **System Level** and **Token Level**. In the Token level, one must identify: the set of token-nets (the P/T nets which describe the structure and the behaviour of the manufacturing system), and the set of token-rules (production "double pushout rules" which describe the reconfigurations that can be applied on the manufacturing system's structure). In the system level, places can be net-places (can contain token-nets) or rule-places (can contain token- rules). The transitions, in the system level, have the ability to trigger the transitions in a token-net, so that they change the token-net marking. In this case, the transitions (of system level) are called **Fire** Transitions. A second ability is to change the token-net structure by the application of a token-rule. In this case, the transitions (of system level) are called **Transform** Transitions.

Identification of token-nets. In the proposed system, three tokens-nets are defined (Figure 9, Figure 10, Figure 11). These three nets represent the three configurations of the system, during its execution. The interpretation of the set of nodes, in these token-nets, is presented on the Table 1.

Identification of the token-rules. In order to simulate the reconfigurations of the system, two productions rules must be defined. A production rule will trigger a reconfiguration processes in the manufacturing system. The construction of these two rules requires the definition of a set of morphisms.

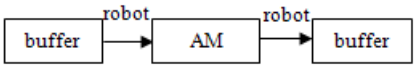

Fig. 6. Flow in MC2

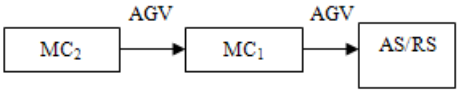

Fig. 7. Reconfiguration 1: Manufacturing of the new product B

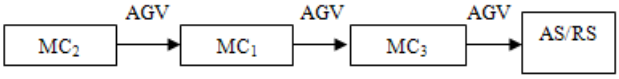

Fig. 8. Reconfiguration 2: introducing MC3

Table 1. Interpretation of the Nodes in the Token-Nets

Node	Interpretation in the manufacturing system
p11	buffer for R1 in MC1
t11	robot and Lathe machine (in MC1)
p12	robot (in MC1)
t12	milling and robot (in MC1)
p13	buffer for final product of MC1
t13, t2, t5	AGVs
p1, p2	Buffer for raw product of MC2
t1	Robot and AM in MC2
p3	Buffer for final product of MC2
p4	AS/RS
p21	buffer for R2 in MC1
t21	robot and Lathe machine (in MC1)
p22	robot (in MC1)
t22	milling and robot (in MC1)
p23	buffer for final product of MC1
p5	Buffer in MC3
t4	CMM
p6	Buffer in MC3

Rule 1: Figure 12 depicts the first rule. The rule contains three components: L: left, I: Interface, and R: Right. We have the first production $p = (L, I, R)$. In this figure: h_1 and h_2 are two morphisms.

On the Figure 13, we depict with more details the two morphisms h_1, and h_2.

Examining the Figure 13, it is easy to see that the two relations h_1 and h_2 are two morphisms. They satisfy the set of relations presented in the paragraph 3.3.

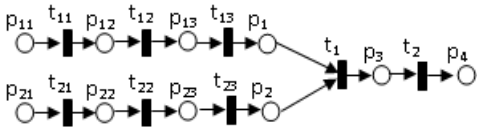

Fig. 9. Token-net (TN_1) for product A manufacturing

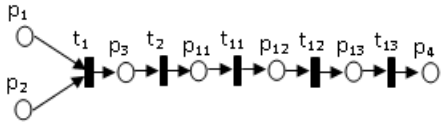

Fig. 10. Token-net (TN_2) for reconfiguration 1

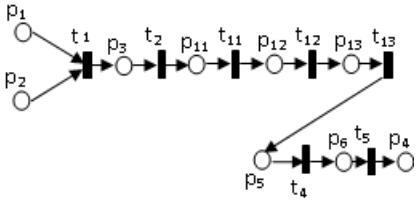

Fig. 11. Token-net (TN_3) for reconfiguration 2

Thus, the first **double pushout rule**, which triggers the first reconfiguration form token-net TN_1 (Figure 9) toward the token-net TN_2 (Figure 10), is depicted on Figure 14.

In Figure 14, the applications h_{11}, h_{12}, c, m, and g are morphisms. The net C is the context of the double pushout. Once the morphism m is defined, the context C can be computed using the definition presented above in the subsection (3.5). The double pushout rule is written as: $r_1 = (p,\ m)$, and the transformation is now written as: $TN_1 \rightarrow^{(p,m)} TN_2$.

The Figure 15 and the Figure 16 present in details the two morphisms m and g. It is also easy to verify that the two relations m and g satisfy the necessary requirements to be morphisms.

Now, we can compute the context C, using the definition presented above in the section (3.5), thus:

1. $T_C = (T_1 \setminus m_T(T_L)) \bigcup m_T(K_{1_T}(T_K)) = \{t_{11}, t_{21}, t_{22}, t_{13}, t_{23}\} \bigcup \{t_{12}\}$
2. $P_C = (P_1 \setminus m_P(P_L)) \bigcup m_P(K_{1_P}(P_K))) = \{p_{11}, p_{22}, p_{23}\} \bigcup \{p_1, p_2, p_3, p_4, p_{12}, p_{13}\}$

The Figure 17 shows the context C.

Rule 2: The Figure 18 shows the second production. The second production is written: $p' = (L',\ I',\ R')$, where: L' for left, I' for Interface, and R' for Right. The two relations: h'_1, h'_2 are two morphisms.

The second double pushout rule, which triggers the second reconfiguration form token-net TN_2 (presented in Figure 10, in the section 4.2.1) toward token-

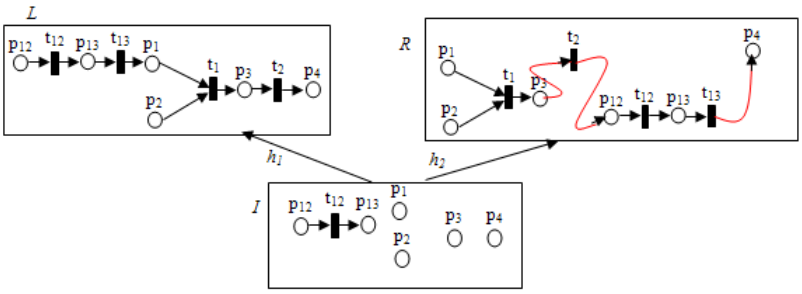

Fig. 12. Production 1

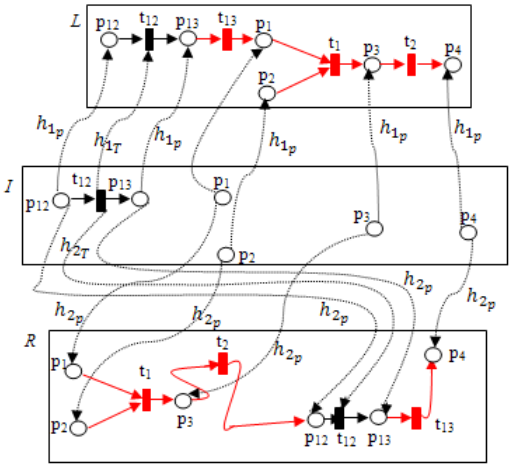

Fig. 13. Morphisms h_1 and h_2

net TN_3 (presented in Figure 11, in the section 4.2.1), is depicted on Figure 19.

In the Figure 19, h'_{11}, h'_{12}, c', m', g' are morphisms, and C' is a context net. Once the morphism m' is defined, the context C' can be computed using the definition presented above in the subsection (3.5). The second double pushout rule is written as: $r_2 = (p', m')$, where $p' = (L', I', R')$, and the second transformation is written as: $TN_2 \rightarrow^{(p',m')} TN_3$.

The system level net. In the system level, we have two kinds of transitions: **Fire-Transitions** which trigger the dynamic behaviour over markings of the token-nets, and **Transform-Transitions** which trigger the reconfiguration behaviour over the structure of the token-nets. Places in the system level can contain two kinds of tokens: token-nets or token-rules. A Fire-Transition takes as parameters: a net N, a transition t from this net, and updates the marking of

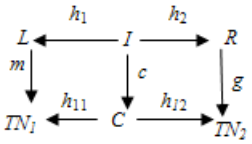

Fig. 14. Double pushout of the rule 1

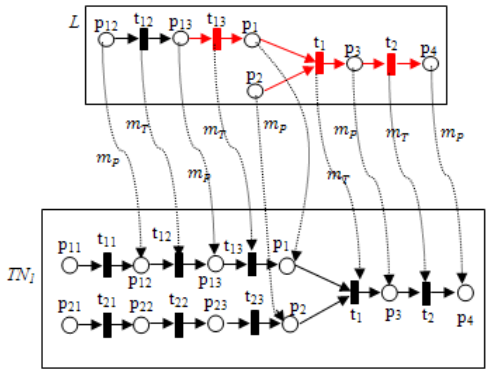

Fig. 15. The morphism m

N by firing t (if this last one is enabled). A Fire-Transition must have a guard $[enabled(t) = true]$. Once fired, this fire-transition produces a new net computed by the function: $fire(N, t)$.

A Transform-Transition takes as parameters: a net N, a rule $r = (p, m)$, and applies this rule to transform N. A transform transition must have a guard $[applicable(N, r)]$. This Transform-Transition produces a new net with a new structure defined by a function: $transform(N, r)$.

In Figure 20, we depict the Reconfigurable Object Net model for the system described in this paper. Each place (a circle) has a name (depicted in the right high side, near the circle), a type (depicted in the right low side, near the circle), and an eventually initial marking (inside the circle). Each transition (a rectangle) has a name (depicted inside the rectangle), and eventually guard (depicted outside the rectangle). An arc links a place to a transition or a transition to a place, and has a label depicted near it.

5 Simulation and verification

The objective of the formal modelling is to understand more the system, to simulate it and to do verification. In this section, we will present the simulation and verification processes, that can be applied in this study. We will use the automatic tool RON-Tool [16] to edit and simulate the reconfiguration process, and

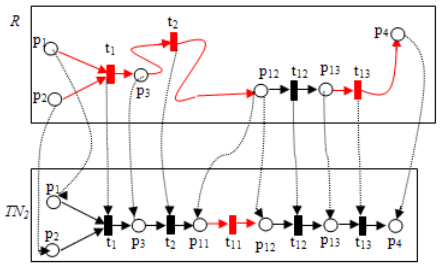

Fig. 16. The morphism g

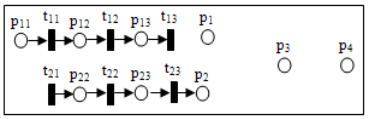

Fig. 17. The context C

the TINA-tool [17] to verify properties of each allowable configuration obtained during the life of the system.

5.1 Simulation

One of the advantages when using the RONs formalism is the ability to simulate the model with the RON-tool [16]. The RONs-tool is free and can be downloaded (with its open source). The current version allows, only, the simulation of the model. The verification of properties is not yet implemented [16]. However, the availability of the source allows the implementation and the specialisation of the verification process by designers. The RONs-tool allows the graphical edition of the model. The user introduces the system net and the object nets, and the set of reconfiguration rules. Figure 21 shows the model of the system edited in the RONs-tool.

The interface presents three windows. The right high window depicts the system level net, the left high window depicts the object net TN_1 in the place np_1, and the low window depicts the transformation rule (token-rule r_1) to be applied on the object net TN_1. The tool can be used to simulate the behaviour of the system level net as well as the behaviour of the object nets. The Transform-Transition 1 has a green colour which means that this transition can be fired.

5.2 Verification

Besides the simulation of the models, we can verify the set of reached configurations by the system. In this section, we propose to use the TINA-tool [17] to verify the P/T nets which model the set of configurations. The TINA-tool (TIme

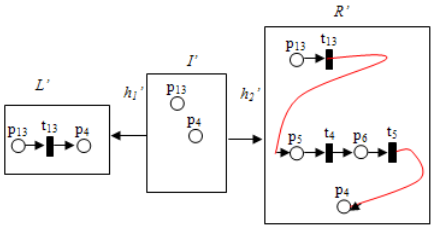

Fig. 18. Production 2

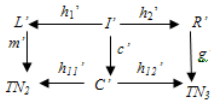

Fig. 19. Double pushout 2

Petri Net Analyser) can compute the state space graph for a Petri Net and verify many properties using this graph. Properties like: reachability, boundedness, liveness can be verified using TINA. Moreover the analysis using the reachability graph, the TINA-tool can do structural analysis of the net using the incidence matrix and invariants. As an example, we present the result of the TINA-tool on the first configuration of the system: Token-net TN_1 (presented in figure 9). figure 22 shows the object net TN_1 and its coverability graph computed using TINA. The proposed initial marking is two tokens in the place p_{11} and two tokens in the place p_{21}.

The coverability graph can be used to verify many properties like: the live transitions and states, the dead transitions and states, the reachable states, etc. According to [18], the reconfiguration of P/T nets based on graph transformation preserve the liveness and the safety properties. Thus, if TN_1 has some live transitions and states then these transitions and states still live in the new configuration TN_2. The designer is not obliged to redo the verification of such properties after reconfiguration.

6 Conclusion

Reconfigurable Manufacturing Systems (RMSs) are systems where the structure changes over time, to satisfy some new requirements or to resolve some structural problems (breakdown machines). Developing these systems and insuring their reliability become an exigency, but also a hard task. The use of formal methods, in particular Petri Nets (PNs), attracts many researchers. Many works applied PNs to specify, simulate, and verify manufacturing systems. The reconfigurability aspect of RMSs presents a challenge for the use of classical Petri nets, therefore some new PNs extensions are proposed to deal with RMSs. In this paper, we have presented an experience where the Reconfigurable Object Nets formalism

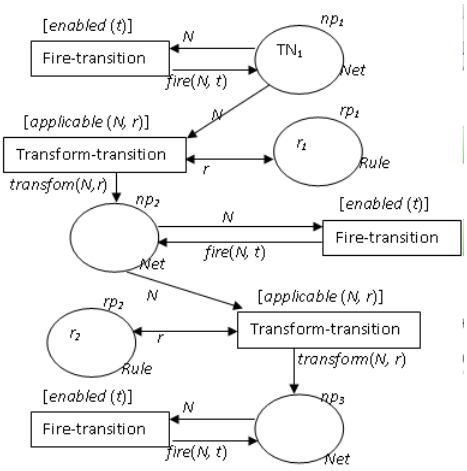

Fig. 20. The RON model of the system

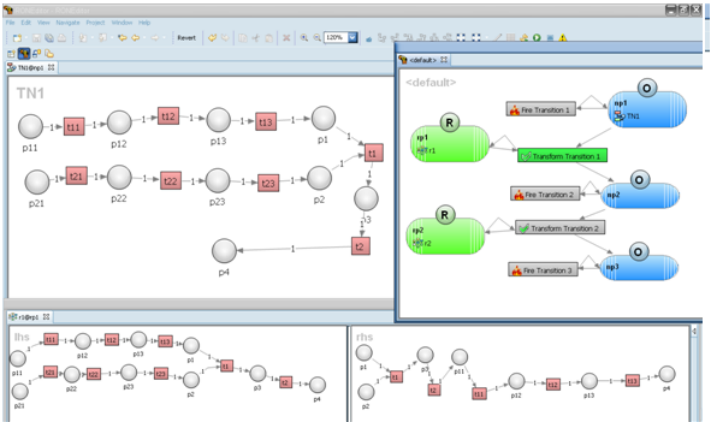

Fig. 21. The model of the system by the RON-tool

[27] is used to specify RMSs. This formalism is based on graph transformation techniques. In this paper, we have examined (step by step) a case study, where the RMS meets two reconfigurations. These two reconfigurations are specified as two transformation-rules to be applied on the structure of the system. We have presented the final model as a RON. After the modelling, we have presented the simulation of the model using the RON-tool [16] (to see the reconfiguration process) and the verification process of one object-net using the TINA-tool [17].

This work opens many perspectives. We propose to develop this work on three levels: (i) enrich the work and develop a concrete approach that can be used in the modelling of RMSs using RONs, (ii) working on the RONs automatic tool (open source), to implement properties verification processes, and finally

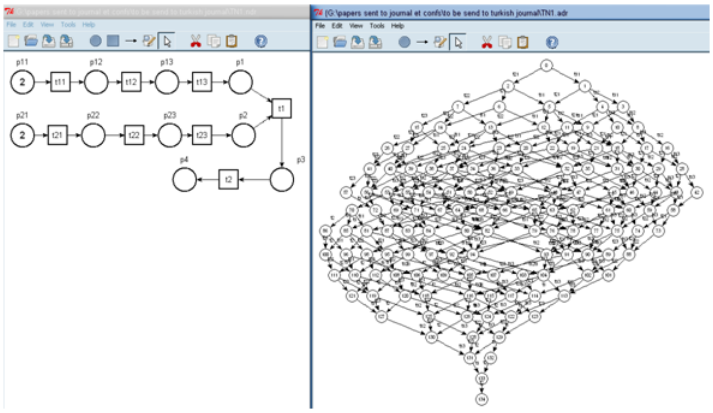

Fig. 22. The token-net TN_1 and its coverability graph

(iii) introducing the time factor in the modelling process. The amelioration of the efficiency (reducing the global time of the manufacturing process) of the system is one of the most motivation of reconfiguration. This efficiency can be verified by the use of temporal model. Time Petri Nets [27] can be used for this purpose; however reconfiguration is not yet well defined for time Petri Nets. One important perspective can be the work on reconfiguration of Time Petri Nets.

References

[1] Serope, K., Schmid, S.: Manufacturing, Engineering & Technology. 6th ed. Prentice Hall (2009)

[2] Mehrabi, MG., Ulsoy, AG., and Koren Y.: Reconfigurable manufacturing systems: key to future manufacturing. Journal of Intelligent Manufacturing. 11, 403-419 (2000)

[3] Katz, R.: Design principles of reconfigurable machines. The International Journal of Advanced Manufacturing Technology. 34, 430-439 (2007)

[4] Korena, Y., Shpitalni, M.: Design of reconfigurable manufacturing systems. Design of reconfigurable manufacturing systems. 29, 130-141 (2010)

[5] Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE. 77, 541-580 (1989)

[6] Valette, R., Cardoso, J., Dubois, D.: Monitoring manufacturing systems by means of Petri nets with imprecise markings. In: IEEE 1989 International Symposium on Intelligent Control; 25-26 September 1989; Albany, NY, USA. pp. 233-238 (1989)

[7] Zhou, MC., Mcdermott, K., Patel, PA.: Petri net synthesis and analysis of a flexible manufacturing system cell. IEEE Transactions on Systems, Man and Cybernetics. 23, 523-531 (1993)

[8] Cheng, CW., Sun, TH., Fu, LC.: Petri-net based modeling and scheduling of a flexible manufacturing system. In: IEEE 1994 International conference on robotics and automation; 8-13 May 1994; San Diego, CA. 513-518 (1994)

[9] Wang, LC.: Object-oriented Petri nets for modeling and analysis of automated manufacturing systems. Computer Integrated Manufacturing Systems. 9, 111-125 (1996)

[10] Valk, R.: Self-modifying nets, a natural extension of petri nets. In: Proceedings of the Fifth Colloquium on Automata, Languages and Programming 1978; 1721 July 1978; Udine, Italy: LNCS 62. 464-476 (1978)
[11] Badouel, E., Llorens, M., Oliver, J.: Modelling Concurrent Systems: Reconfigurable Nets. In Proceeding of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03) 2003, 23-26 June 2003; Las Vegas, Nevada, USA. pp. 1568-1574 (2003)
[12] Valk, R.: Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In: Applications and Theory of Petri Nets ICATPN'98; June 1998; Lisbon, Portugal: LNCS 1420. pp. 1–25 (1998)
[13] Lomazova, IA.: Nested Petri Nets: Multi-level and Recursive Systems. Fundamenta Informaticae. 47, 283–293 (2001)
[14] Asperti, A., Busi, N.: Mobile Petri Nets. Mathematical Structures in Computer Science. 19, 1265–1278 (2009)
[15] Hoffmann, K., Ehrig, H., Mossakowski, T.: High-Level Nets with Nets and Rules as Tokens. In: Proceedings of the 26th international conference on Applications and Theory of Petri Nets; 20-25 June 2005; Miami, USA: LNCS 3536. pp. 268288 (2005)
[16] Biermann, E., Ermel, C., Hermann, F., Modica, TA.: Visual Editor for Reconfigurable Object Nets based on the ECLIPSE Graphical Editor Framework. In: Proceeding of the 14th Workshop on Algorithms and Tools for Petri Nets (AWPN2007); 20-21 September 2006; Koblenz, Germany. pp. 1-6 (2007)
[17] Berthomieu, B., Ribet, PO., Vernadat, F.: The tool TINA construction of abstract state spaces for Petri nets and time Petri nets. International Journal of Production Research. 42, 2741–2756 (2004)
[18] Ehrig, H., Padberg, J.: Graph Grammars and Petri Net Transformations. In: Desel, Jrg; Reisig, Wolfgang; Rozenberg, Grzegorz, editors. Lectures on Concurrency and Petri Nets: Advanced Course PNT, 2004; Springer Berlin Heidelberg. pp. 496-536 (2004)
[19] Meng, X.: Modeling of reconfigurable manufacturing systems based on colored timed object-oriented Petri nets. Journal of Manufacturing Systems. 29, 81-90 (2010)
[20] Julia S., de Oliveira, FF., Valette, R.: Real time scheduling of Workflow Management Systems based on a p-time Petri net model with hybrid resources. Simulation Modelling Practice and Theory. 16, 462-482 (2008)
[21] Cunha de Aguiar, AJ., Villani, E., Junqueira, F., Coloured Petri nets and graphical simulation for the validation of a robotic cell in aircraft industry. Robotics and Computer-Integrated Manufacturin. 27, 929-941 (2011)
[22] Wu, N., Zhou, M.: Intelligent token Petri nets for modelling and control of reconfigurable automated manufacturing systems with dynamical changes. Transactions of the Institute of Measurement and Control. 33, 9-29 (2009)
[23] Li, J, Dai, X, Meng, Z, Dou, J and Guan, X, (2009). Rapid design and reconfiguration of Petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams. Computers & Industrial Engineering, 57(4) 14311451.
[24] Lejri, O., Tagina, M.: Hybrid Reconfigurable Petri Nets for modelling Hybrid Reconfigurable Manufacturing Systems. Journal of Studies on Manufacturing. 1, 75-84 (2011)
[25] Prange, U., Ehrig, H., Hoffmann, K., Padberg, J.: Transformations in Reconfigurable Place/Transition Systems. In: Degano P, Nicola RD, Meseguer J, editors. Concurrency, Graphs and Models 2008, LNCS 5065, Springer Berlin Heidelberg. pp. 96-113 (2008)

[26] Biermann, E., Modica, T.: Independence Analysis of Firing and Rule-based Net Transformations in Reconfigurable Object Nets. Electronic Communications of the EASST. 10 1-13, (2008)
[27] Merlin, PM., Farber, DJ.: Recoverability of communication protocols: Implications of a theoretical study. IEEE Transactions on Communication. 24, 10361043 (1976)
[28] Valk, R.: Petri Nets as Token Objects: An Introduction to Elementary Object Nets, Proc. of the International Conference on Application and Theory of Petri Nets, LNCS 1420, pages 125, (1998)
[29] Valk, R.: Concurrency in Communicating Object Petri Nets, In G. Agha, F. de Cindio, and G. Rozenberg, editors, Concurrent Object-Oriented Programming and Petri Nets, LNCS 2001, pages 164–195. Springer. (2001)
[30] Ede, M., Hoffmann, K., Oelker, G., Padberg, J.: RECONNET: A Tool for Modeling and Simulating with Reconfigurable Place/Transition Nets, Electronic Communications of the EASST 7th International Workshop on Graph Based Tools, (GraBaTs 2012), Vol. 54, (2012).
[31] Badouel, E., Oliver, J.: Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes within Workow Systems, INRIA report N 3339, (1998)
[32] U2 Partners, Unified Modeling Language: Superstructure, version 2.0, 3rd revised submission to OMG RFP ad/00-09-02, http://www.omg.org/cgi-bin/doc?ad/2003-04-01, April 2003.
[33] Abid, K., Kahloul, L., Mous, LH., Kazar, O.: Formal Specification of a Mobile Agent Based Maintenance for Manufacturing Systems. In the 7th International Workshop on Verification and Evaluation of Computer and Communication Systems. Florence, Italy, November 21-22, 2013.
[34] Richta, T., Janouek, V., Ko, R.: Petri Nets-Based Development of Dynamically Reconfigurable Embedded Systems. In proceeding of Petri Nets and Software Engineering. pp. 203218. (2013).
[35] Aalst, VD. Hee, KV.: Workflow Management: Models, Methods, and Systems. IT press, Cambridge, MA. (2002).
[36] Valk, R.: Object Petri Nets Using the Nets-within-Nets Paradigm. In: Jrg Desel, Wolfgang Reisig, and G.R. (ed.) Advances in Petri Nets: Lectures on Concurrency and Petri Nets, volume 3098 of Lecture Notes in Compu ter Science. pp. 819-848. Springer-Verlag, Berlin, Hei- delberg, New York, New York, USA. (2004)
[37] Zhang, L., Rodrigues, B.: Modeling Reconfigurable Manufacturing Systems With Colored Timed Petri Nets. International Journal of Production Research. 47, 4569-4591 (2009)
[38] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Brussel, HV.: Reconfigurable manufacturing systems. CIRP Annals - Manufacturing Technology. 48, 527-540 (1999)