PRODUCTION CONFIGURATION OF PRODUCT FAMILIES
An Approach based on Petri Nets

Lianfeng Zhang, Brian Rodrigues
Department of Operations, University of Groningen, Landleven 5, Groningen, The Netherlands
Lee Kong Chian School of Business, Singapore Management University, Singapore

Jannes Slomp, Gerard J. C. Gaalman
Department of Operations, University of Groningen, Landleven 5, Groningen, The Netherlands

Keywords: Production configuration, Timed Petri nets, Colored Petri nets, Transition synchronization.

Abstract: Configuring production processes for product families has been acknowledged as an effective means of dealing with product variety while maintaining production stability and efficiency. In an attempt to assist practitioners to better understand and implement production configuration, we study the underlying logic for configuring production processes for product families by means of dynamic modelling and visualization. Accordingly, we develop a formalism of nested colored timed Petri nets (PNs) to model production configuration. To cope with the modelling difficulties resulting from the fundamental issues in production configuration, three types of nets, namely process nets, assembly nets and manufacturing nets, together with a nested net system are defined by integrating the principles of colored PNs, timed PNs and nested PNs. An industrial example of electronics products is used throughout the whole paper to demonstrates how the proposed formalism is applied to specify production processes at differently levels of abstraction to achieve production configuration.

1 INTRODUCTION

Facing a high variety of customized products often required in small quantities and with short delivery lead-times, manufacturing companies nowadays strive to provide quickly diverse products at low costs so as to survive the intense global competition. It has been recognized that the key for companies to achieving efficiency in the resulting high variety production lies in an ability to maintain it to be as stable as possible (Martinez et al., 2000). In this respect, process configuration contributes to obtaining manufacturing stability by generating similar manufacturing processes for parts (Schierholt, 2001). It, *de facto*, is an alternative computer-aided process planning and thus does not lend itself to achieve efficiency in producing product families, where both assemblies and parts are involved. In response to the limited research in product family production , production configuration has been proposed for companies to produce product variety while achieving mass production efficiency (Zhang, 2007). The rationale is to plan and utilize similar, yet optimal, production processes as these existing on shop floors to fulfil customized products.

Production configuration entails a conceptual structure and overall logical organization of producing a family of individualized products. It provides a generic umbrella to capture and utilize commonality, within which each new product fulfilment is instantiated and extended, and thereby anchoring planning to a common structure of a product family (Zhang et al., 2007). Underpinned by the common structure, production configuration involves such common elements as operations and processes concepts for parts and assemblies, operations and their precedence, and manufacturing resources. Also involved are diverse optional process elements in accordance with optional parts and assemblies. Production configuration entails a process of 1) selecting proper elements for given products; 2) subsequently arranging the selected elements into production processes; and 3) finally evaluating the configured multiple alternatives to

Zhang L., Rodrigues B., Slomp J. and J. C. Gaalman G. (2008).
PRODUCTION CONFIGURATION OF PRODUCT FAMILIES - An Approach based on Petri Nets.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 5-11
DOI: 10.5220/0001669400050011

determine appropriate ones. In such selection, arrangement and evaluation processes, a number of configuration rules, i.e., constraints or restrictions imposed by manufacturing capabilities, capacity limits, desired objectives that measure productivity and efficiency, etc. must be satisfied. Consequently, decision making in production configuration is deemed to be very complex.

With an attempt to assist practitioners to better understand and implement production configuration, in this study, we tackle the underlying logic for configuring production processes for product families. In view of the potential of dynamic modelling and visualization in shedding light on process logic, we propose to model production configuration graphically. In relation to fundamental issues in production configuration (Zhang, 2007), the resulting difficulties in modelling production configuration have been identified. They include 1) handling a high product and process variety; 2) accommodating many process variations resulting from design changes; 3) dealing with configuration at different abstraction levels; 4) satisfying a number of constraints/restrictions describing, for example, the connections between product and process elements; and 5) selecting process elements/ processes from multiple alternatives.

Owing to their executability, graphical representation and mathematic support, Petri nets (PNs) have been well recognized as a powerful modelling, simulation and evaluation tool for complex flows and processes (Peterson, 1981). A number of extensions have been made to classic PNs in order to satisfy different modelling requirements. Among all, colored PNs (CPNs; Jensen, 1995), timed PNs (TPNs; Ramachandani, 1974) and nested PNs (NPNs; Lomazova, 2000) are of particular interest in this study. CPNs are able to provide a concise, flexible and manageable representation of large systems by attaching a variety of colors to tokens. By including timing, TPNs can capture systems physical behaviours through assuming specific durations for various activities. By considering PNs as tokens, NPNs is expected to model large and complex system through refinement and abstraction.

In this paper, we apply PN techniques to model production configuration by taking into account the difficulties imposed by the fundamental issues. Accordingly, a formalism of NCTPNs (nested colored timed PNs) has been developed by integrating the basic principles of CPNs, TPNs and NPNs. Product/process variety and configuration constraints are handled by attaching various data describing product items and process elements to tokens, resulting colored tokens. A reconfiguration mechanism is incorporated into the modelling formalism to cope with process variations. Timing is introduced to facilitate the selection of proper machines, operations and processes. Moreover, a multilevel net system is further developed based on net nesting in NPNs to tackle explicitly the granularity issue in production configuration.

The next section introduces an industrial example of vibration motors for mobile phones, which is used throughout the rest of this paper to demonstrate the proposed formalism and the modelling of production configuration.

2 INDUSTRIAL EXAMPLE

The industrial example adopted is the production of vibration motors for mobile phones. The increasing variations in mobile phones design lead to large numbers of customized vibration motors to be produced. Together with other factors, e.g., short delivery lead times, the high variety of vibration motors complicate the planning of their production processes.

Modelling production configuration of a vibration motor family using the proposed NCTPNs is presented in this study. Figure 1 shows the common product structure of the motor family. This motor family assumes several part types, as shown in the figure. According to the design requirements, individual motors may not contain a specific variant from each part family. In other words, not all part types are assumed by each individual motor variant. While the company produces all assembly families and final motors, among all they manufacture several part families, e.g., Ba family, Bb family, Tl family.

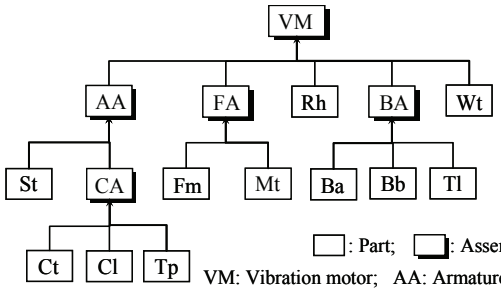

Figure 1: The common product structure.

3 NET DEFINITIONS

Production configuration entails a process of configuring a complete production process for a product along with the product hierarchy, i.e., at different levels of abstraction. Being located at the highest level, the more abstract production process involving the product's immediate child items only is configured first. Subsequently, the assembly processes and/or manufacturing processes for producing these immediate child items are configured at the second level, and so on. The manufacturing processes for parts at the lowest level of each branch of the hierarchy are configured at last. Bearing in mind the similarities embedded in processes of any type, e.g., adoption of buffers and alternative machines, involvement of WIPs, we define a basic PN structure first, which reflects the generalized common process elements that are assumed by different types of nets defined in the proposed formalism.

3.1 Basic Net Structure

Definition 1: A basic PN structure is a directed bipartite graph $G = (P, T, A, h)$, where

$P = P^B \cup P^I \cup P^R \cup P^{CR}$ is a finite set of places with 4 disjoint subsets: P^B representing buffers (be it for raw materials, parts, assemblies, or final products), P^I items ready to be processed (be it a raw material, part, assembly, or WIP), P^R machines, and P^{CR} the conceptual machines for alternative machines that can complete same tasks using different operations;

$T = T^L \cup T^R \cup T^T, P \cap T = \phi$ is a finite set of transitions with 3 disjoint subsets: T^L denoting a set of logical transitions, T^R a set of reconfigurable transitions, and T^T a set of timed transitions;

$A \subseteq P \times T \cup T \times P$ is a finite set of arcs that connect places/transitions to transitions/places; and

$h \subseteq P^{CR} \times T^R, h \cap A = \phi$ is a finite set of inhibitor arcs that connect a conceptual machine place to a reconfigurable transition. $h(p,t) = 1, \forall p \in P^{CR}, t \in T^R$ indicates that there is a token in the conceptual machine place; and the associated reconfigurable transition is disabled and cannot fire. When $h(p,t) = 0$, no token resides in the conceptual machine place; and the associated reconfigurable transition can fire if it is enabled.

Attempting to capture and model multiple alternative machines in relation to same tasks, P^{CR} is defined in the formalism to represent the corresponding conceptual machines in addition to P^R, which is defined to denote specific machines. In conjunction with P^{CR}, h and T^R are introduced to model the situations, where multiple machines can perform same tasks and only one is used eventually. The firing of T^R leads to the reconfiguration of proper machines. In this way, P^{CR}, T^R and h can address process variations in system models without rebuilding new ones when machines are added and/or removed.

T^L is defined to capture the logic of the system running. Their firing indicates the satisfaction of preconditions of operations, for example, the presence of material items and machines to be used. T^T is defined to represent operations, which take certain time durations to complete. Thus, the firing of timed transitions incurs time delays. Both logic and reconfigurable transitions are untimed. Their firing is instantaneous and takes 0 time delay.

Figure 2 shows a basic PN structure and the corresponding graphical formalism. Based on the basic PN structure, 3 types of PNs, namely manufacturing nets (*MNets*), assembly nets (*ANets*) and a process net (*PNet*) are defined to address the granularity issue in production configuration. These nets are defined as a type with a marking, with each type following the basic PN structure and includes additional information.

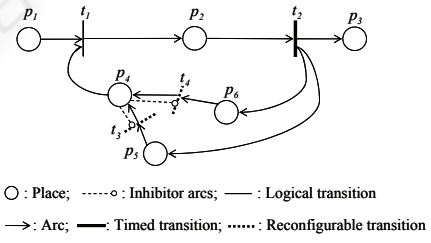

Figure 2: A basic PN structure.

3.2 Manufacturing Net (*MNet*)

Definition 2: A manufacturing net is defined as a tuple $MNet = (ToM, \mu)$, where

MNet is a manufacturing net representing the processes of manufacturing a part family;

$ToM = (G, \Sigma^M, \alpha, \beta, E, \tau)$ is a manufacturing type with

– G is the basic PN structure;

– Σ^M is a finite set of color sets or token types;

– $\alpha : P \mapsto M_{\Sigma^M}$ is a color assignment function that maps a place, p, to a set of colors, $\alpha(p)$;

– $\beta : T \mapsto (\Sigma^M) \times (\Sigma^M \cup \{\varepsilon\})$ is a color assignment function that maps a transition, t, to a set of color pairs, $\beta(t)$ such that each pair declares the transition, t, must be synchronized externally with transitions of the net where this manufacturing net resides, except if the pair is (Σ^M, ε), indicating no synchronization is required;

– $E : A \times \Sigma^M \mapsto \vee M_{\Sigma^M} \cup \vee(\wedge M_{\Sigma^M} \to @M_{\Sigma^M} + \tau)$ is an arc expression function that defines the timed and untimed arc expressions for arcs with respect to transition colors; $\vee / \wedge$ denote *Exclusive OR* (*XOR*)/*AND* relationships; and $\to$ represents "*if–then*";

– $\tau \in \Re^+$ is a set of positive real numbers representing time delays;

$\mu : P \mapsto M_{\alpha(p)}, \forall p \in P$ is a marking function specifying the distribution of colored tokens in all places of an *MNet* .

While tokens in idle machine places are defined to represent machines, tokens in busy machine places are defined according to the items to be produced. When there is a token residing in the conceptual machine place, the conceptual machine is instantiated to the specific machine represented by the token. Cycle times are used to accommodate selection of proper processes elements. Thus, time delays, τ, are defined in the formalism to represent operations cycle times.

To cope with the difficulties in modelling diverse cycle times, an arc expression function, E, is introduced in the formalism. It defines both timed arc expressions and untimed arc expressions. A timed arc expression is a set of antecedent-consequent statements with *XOR* relationships. It specifies input items, machines to be used, output items and the cycle times to be incurred. Untimed arc expressions are defined to specify 1) the input tokens for firing any transitions; and 2) output tokens after firing timed and reconfigurable transitions.

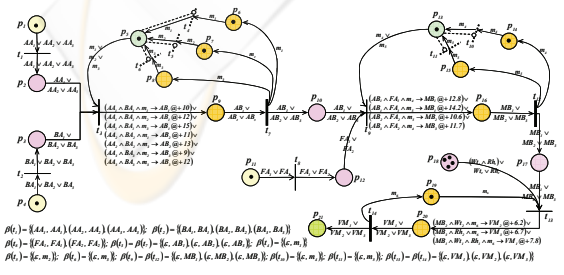

Figure 3: The *PNet* of the motor family.

By following the common routing of the motor family, the *PNet* is constructed, as shown in Figure 3. In the *PNet*, the involved items include AAs, FAs, BAs, ABs (abassies; WIPs formed by AAs and BAs), MBs (mainbodies; WIPs formed by ABs and FAs) and VMs. The places, represented system elements and contained colored tokens/lower level nets are listed in Table 1 (not exhaust due to place constraints). The color pairs assigned to each transaction are shown in the figure. For illustrative simplicity, the colored tokens in relation to 3 motor variants (VM_1, VM_2 and VM_3) are given in Figure 3 and Table 1, and in the following figures and tables as well. Take the timed arc expression of arc (t_{13}, p_{20}) as an example. It shows that VM_1 assumes both Wt and Rh; VM_2 contains a Wt; and VM_3 includes an Rh. Also shown are the machines used and the corresponding cycle times.

Table 1: The places, represented system elements and colored tokens/nets.

Places	System Elements	Nets/Colored Tokens
$p_{1/4/11}$	Input buffers for AAs/Bas /FAs	*ANets* for producing assembly families AA/BA/FA
p_2	AAs are ready to be processed	$\alpha(p_2) = \{AA_1, AA_2, AA_3\}$
p_3	BAs are ready to be processed	$\alpha(p_2) = \{BA_1, BA_2, BA_3\}$
…	…	…
p_{21}	A buffer for final motors	$\alpha(p_{21}) = \{VM_1, VM_2, VM_3\}$

3.3 Assembly Net (*ANet*)

Definition 3: An assembly net is defined as a tuple $ANet = (ToA, \mu)$, where

ANet is an assembly net representing the processes of producing a family of assembly variants;

$ToA = (G, SoTma, \Sigma^A, \alpha, \beta, E, \tau)$ is an assembly type with

– G is the basic PN structure;

– *SoTma* is a set of manufacturing types and assembly types;

– Σ^A is a finite set of color sets or token types;

– $\alpha : P \mapsto SoTma \cup M_{\Sigma^A}$ is an assignment function that maps a place, p, to manufacturing or assembly types or a set of colors;

– $\beta : T \mapsto (\Sigma^A \cup \{\varepsilon\}) \times (\Sigma^A) \times (\Sigma^A \cup \{\varepsilon\})$ is an assignment function that maps a transition, t, to a set of color triples, $\beta(t)$, such that each triple declares

the transition, t, must be synchronized 1) internally with transitions of the nested net; and 2) externally with transitions of the host net where this assembly net is contained, except if the triple is $(\varepsilon, \Sigma^A, \varepsilon)$;

– $E: A \times \Sigma^A \mapsto \vee M_{\Sigma^A} \cup \vee(\wedge M_{\Sigma^A} \to @M_{\Sigma^A} + \tau)$ is an arc expression function that defines the timed and untimed arc expressions for arcs with respect to transition colors; and

– $\tau \in \Re^+$ is a set of positive real numbers representing time delays;

$\mu: P \mapsto M_{\alpha(p)}, \forall p \in P$ is a marking function specifying the distribution of colored tokens, manufacturing and assembly nets in all places of an *ANet*.

Unlike raw buffer places in an *MNet*, buffer places in an *ANet* are defined for the immediate child items of assemblies. Tokens in such part/subassembly buffers are *MNets* / *ANets* of child items. The assignment function, α, specifies the allocation of such *MNets* / *ANets* to buffer places and colored tokens to other places.

Same as specifying the *PNet*, four *ANets* have been constructed for 4 assembly families of the motor family in Figure 1. Figure 4 shows the *ANet* of BA family. The color triples assigned to each transition are also shown. According to these colored triples, transitions of this *ANet* fires autonomously, or simultaneously with either the transitions of the nested nets residing in places p_{22}, p_{25} and p_{31} or the transitions of the *PNet* in Figure 3. Table 2 lists the set of places, represented system elements and colored tokens/ *MNets*.

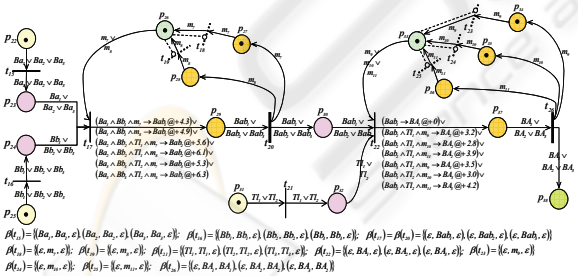

Figure 4: The *ANet* of the bracketassy family.

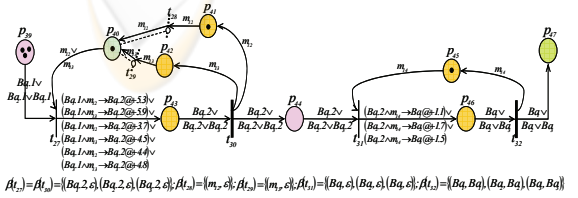

Figure 5: The *MNet* of the bracket a family.

Table 2: The places, represented elements and colored tokens/nets.

Places	System Elements	Nets/Colored Tokens
$p_{22\,/\,25\,/\,31}$	Input part buffers for Bas/Bbs/Tls	*MNets* for producing part families Ba/Bb/Tl
p_{23}	Bas ready to be processed	$\alpha(p_{23}) = \{Ba_1, Ba_2, Ba_3\}$
p_{24}	Bbs ready to be processed	$\alpha(p_{24}) = \{Bb_1, Bb_2, Bb_3\}$
….	…	…
p_{37}	Alternative mach. processing Babs&Tls	$\alpha(p_{37}) = \{BA_1, BA_2, BA_3\}$
p_{38}	A buffer for BAs	$\alpha(p_{38}) = \{BA_1, BA_2, BA_3\}$

3.4 Process Net (*PNet*)

Definition 4: A process net is defined as a tuple $PNet = (ToP, \mu)$.

PNet is a process net representing more abstract production processes of producing a family of end-products. $ToP = (G, SoTma, \Sigma^P, \alpha, \beta, E, \tau)$ is a process type (a special assembly type) with symbols carrying similar meanings as with an *ANet*, except $\beta: T \mapsto (\Sigma^P \cup \{\varepsilon\}) \times (\Sigma^P)$, which is defined as an assignment function that maps a transition, t, to a set of color pairs, $\beta(t)$, such that a pair declares the transition, t, must be synchronized internally with transitions of the nested nets, except if the pair is (ε, Σ^P). Essentially, a *PNet* is a special kind of *ANets*. It involves only the child items at the immediate lower level of the end-products in the BOM structures.

In a similar way, *MNets* have been constructed in accordance with part families produced in house. Figure 5 shows the *MNet* of Ba family. Table 3 lists the places, represented system elements and colored tokens.

4 NESTED NET SYSTEM

4.1 Nested Net System

Definition 5: A multilevel nested net system is defined as a triple, $MlNNS = (PNet, M, A)$, where

MlNNS is the multilevel nested net system for modelling the complete production processes of end-products;

Table 3: The places, represented elements and colored tokens/nets.

Places	System Elements	Colored Tokens
p_{39}	A raw material buffer for Bas	$\alpha(p_{39}) = \{Ba_1.1, Ba_2.1, Ba_3.1\}$
p_{40}	A conceptual machines for manufacturing Bas	$\alpha(p_{40}) = \{m_{12}, m_{13}\}$
$p_{41/42}$	Alternative idle machines for manufacturing Bas	$\alpha(p_{41}) = \{m_{12}\} / \alpha(p_{42}) = \{m_{13}\}$
…	…	…
p_{46}	The second machine processing Ba WIPs	$\alpha(p_{46}) = \{Ba_1, Ba_2, Ba_3\}$
p_{47}	A buffer for Bas	$\alpha(p_{47}) = \{Ba_1, Ba_2, Ba_3\}$

PNet is the process net describing the abstract production processes of end-products;

$M = \{MNet_i\}_{N^M}$ is a finite set of *MNets*; and

$A = \{ANet_i\}_{N^A}$ is a finite set of *ANets*.

Performing as an abstraction mechanism, *MlNNS* facilitates the configuration of processes with right amount of details. Within *MlNNS*, the highest level is the *PNet*, while a number of *MNets* and *ANets* are located at the second level. Each of these nets provides more details for processes of input items involved in the *PNet*. At the lowest level of each path, all nets are *MNets*, whilst a mixture of *MNets* and *ANets* can be found at any arbitrary level.

For the motor family, based on the constructed *MNets*, *ANets* and *PNet*, the multilevel system of production configuration is obtained, as shown in Figure 5.

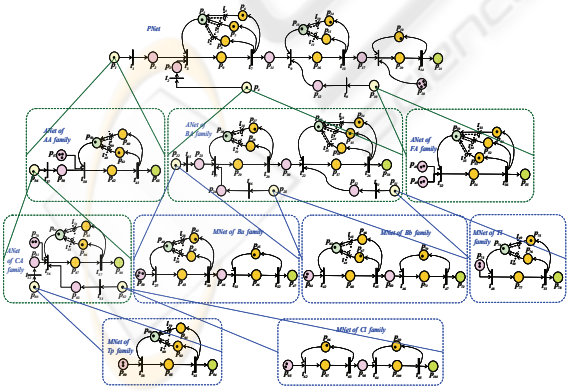

Figure 5: The multilevel nested net system of production configuration.

As shown in the figure, the *PNet* is at the highest level; 3 *ANets* for producing assembly families AA, BA and FA are located at the second level; 1 *ANet* for producing CA family and 3 *MNets* for manufacturing part families Ba, Bb and Tl are at the third level; and 2 *MNets* for fabricating part families Tp and Cl are located at the lowest level. Each lower level nested net is linked with the corresponding tokens residing in the places of the higher level nets.

4.2 System Evolution

Complying with production practice, in the *MlNNS*, *MNets* nest in places of *ANets* and/or *PNet*; and *ANets* nest in places of higher level *ANets* and/or *PNet*. Consequently, configuration of a complete production process for an end-product necessitates interaction of the nested nets and the host nets. In this study, transition synchronization is introduced to enable interaction of nets at different levels. Transition synchronization is declared by the color pairs/triples of net transitions. More specifically, two or more transitions with same color pairs and/or triples must be synchronized, i.e., fire simultaneously. To enable transition synchronization, several rules for enabling and firing transitions are defined in the net system.

Rule 1: A transition, t, of an *MNet* is enabled with respect to a color pair, cp, and fires autonomously without additional conditions if $\forall p, E((p,t),cp) \subseteq \mu(p)$ and $cp = (\Sigma^M, \varepsilon)$.

Rule 2: A transition, t, of an *MNet* is enabled and fires simultaneously with an output transition of the place, where the *MNet* is nested, of the higher level *ANet* or *PNet* if $\forall p, E((p,t),cp) \subseteq \mu(p)$ and $cp = (\Sigma^M, \Sigma^M)$.

Rule 3: A transition, t, of an *ANet* is enabled with respect to a color triple, ct, and fires autonomously without additional conditions if $\forall p, E((p,t),ct) \subseteq \mu(p)$ and $ct = (\varepsilon, \Sigma^A, \varepsilon)$.

Rule 4: A transition, t, of an *ANet* is enabled and fires simultaneously with a transition bearing a $cp = (\Sigma^A, \Sigma^A)$ or $ct = (\varepsilon, \Sigma^A, \Sigma^A)$ of the lower level *MNet* or *ANet* that nests in an input place of t, if $\forall p, E((p,t),ct) \subseteq \mu(p)$ and $ct = (\Sigma^A, \Sigma^A, \varepsilon)$.

Rule 5: A transition, t, of an *ANet* is enabled and fires simultaneously with an output transition of a place, where the *ANet* is nested, of the higher level *ANet* or *PNet*, if $\forall p, E((p,t),ct) \subseteq \mu(p)$ and $ct = (\varepsilon, \Sigma^A, \Sigma^A)$.

Rule 6: A transition, t, of a *PNet* is enabled with respect to a color pair, cp, and fires

autonomously without additional conditions if $\forall p, E((p,t),cp) \subseteq \mu(p)$ and $cp = (\varepsilon, \Sigma^P)$.

Rule 7: A transition, t, of a *PNet* is enabled and fires simultaneously with a transition bearing a $cp = (\Sigma^M, \Sigma^M)$ or $ct = (\varepsilon, \Sigma^A, \Sigma^A)$ of the lower level *MNet* or *ANet* that nests in an input place of t, if $\forall p, E((p,t),cp) \subseteq \mu(p)$ and $cp = (\Sigma^P, \Sigma^P)$.

While the firing of transitions of any net does not provoke the transfer of nested nets, that is, the nested nets remain in the same places before and after simultaneous transition firing, the tokens, other than the nested nets, are created and removed as follows:

Rule 8: The firing of an enabled transition, t, in nets at all levels modifying the marking through 1) generating tokens in the output places by following $E((t,p),cp)$ or $E((t,p),ct)$; and 2) removing tokens in the input places by following $E((p,t),cp)$ or $E((p,t),ct)$.

5 APPLICATION RESULTS

For the 3 motor variants: VM_1, VM_2 and VM_3 in Figure 3 and Table 1, more than one production process consisting of different combination of machines is feasible to fulfil each motor variant. In this regard, minimizing the completion time of the last operation of producing three motor variants is chosen as the production objective and, the selection of appropriate ones is made based on this objective. In the application, we adopt the shortest delay times to fire transitions, which are enabled by alternative colored tokens. Table 5 list the corresponding processes with respect to the selected machines. Due to the space constraints, this table is truncated.

Table 5: The configured production processes for the 3 motor variants.

Machine(Operation; Output item)		
VM_1	VM_2	VM_3
m_{13} (Manufacturing operation; Ba_1)	m_{13} (Manufacturing operation; Ba_2)	m_{12} (Manufacturing operation; Ba_3)
m_{14} (Manufacturing operation; Ba_1)	m_{14} (Manufacturing operation; Ba_2)	m_{14} (Manufacturing operation; Ba_2)
…	…	…
m_5 (Assembly operation; MB_1)	m_5 (Assembly operation; MB_2)	m_4 (Assembly operation; MB_3)
m_6 (Assembly operation; VM_1)	m_6 (Assembly operation; VM_2)	m_6 (Assembly operation; VM_3)

6 CONCLUSIONS

This study addresses the logic for configuring production processes of product families with an attempt to assist practitioners to better understand and implement production configuration. In view of the significance of PN techniques in modelling complex systems/processes, we propose a formalism of NCTPNs to model production configuration. By integrating the basic principles of several well-defined PN extensions, the NCTPNs are able to capture and reflect the complex process of configuring complete production processes for end products. The industrial example has shown the potential of the NCTPNs formalism to reveal the logic of production configuration.

REFERENCES

Dash, S., Kalagnanam, J., Reddy, C., Song, S.H., 2007. Production design for plate products in the steel industry, *IBM Journal of Research & Development,* 51, 354-362.

Jensen, K., 1995. *Colored Petri nets: Basic concepts, analysis methods and practical Use*. Vol. 2, New York, Springer.

Lomazova, I.A., 2000. Nested Petri nets: A formalism for specification and verification of multi-agent distributed systems, *Fundamenta Informaticae*, 43, 195-214.

Martinez, M.T., Favrel, J., Ghodous, P., 2000. Product family manufacturing plan generation and classification, *Concurrent Engineering: Research and Applications*, 8, 12-22.

Peterson, J.L., 1981. *Petri net theory and the modelling of Systems*, Prentice-Hall, Englewood Cliffs, N.J.

Ramachandani, C., 1974. Analysis of asynchronous concurrent systems by timed Petri nets, Technical Report MAC TR 120, MIT, Cambridge.

Schierholt, K., 2001. Process configuration: Combining the principles of product configuration and process planning, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15, 411-424.

Zhang, L., 2007. Process Platform-based Production Configuration for Mass Customization, PhD Dissertation, Division of Systems and Engineering Management, Nanyang Technological University, Singapore.

Zhang, L., Jiao, J., Helo, P., 2007. Process platform representation based on unified modelling language, *International Journal of Production Research*, 45, 323-350.