Model Transform Based on a Kind of Transition Subnet

Zhijian Wang
Information Science School, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: zjian@gdcc.edu.cn

Dingguo Wei
Information Science School, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: weidg@scnu.edu.cn

Cheng Xu
1 Guangdong Key Lab of Electronic Commerce, Guangdong University of Business Studies, Guanzhou, P.R.China
2 School of Computer and Communication, Hunan University, Changsha, P.R.China
Email: chengxu@public.cs.hn.cn

Abstract—Transition refinement and subnet abstraction are key approaches to realize modularization and hierarchical modeling in a Petri nets model; using these two methods, the Petri nets based structural design becomes possible. Model transforming technologies using a kind of transition subnet – the engineering subnet, are presented in this paper; the operations of subnet abstraction and transition refinement are defined. The research shows that, under certain preconditions, important prosperities such as boundedness, safety, deadlock free and reversibility are reserved in subnet abstraction transformations; when a transition is refined using an engineering subnet, prosperities including boundedness, deadlock free and reversibility keeps reserved, but safety is not certainly to be effective any longer. The paper also shows that after the transformation a model can keep the same interfaces and similar actions as before by using engineering subnet method. Petri nets based modeling in complex systems becomes feasible and effective with the help of engineering subnet.

Index Terms—Petri nets, subnet, transition, modeling, abstraction, refinement

I. INTRODUCTION

Petri nets reflect system properties such as parallelism, synchronization and resource sharing intuitively, and provide powerful mathematical analyzing ability. Today Petri nets have become a widely used tool for system simulation and analysis. But modeling complex systems using common Petri nets usually brings the problem of state explosion, which makes system modeling become a difficult or even impossible problem in such occasion. Extended Petri nets provide one way to solve the above problem, Colored Petri Nets(CPN) and Object Oriented Petri Nets(OOPN) are presented to enhance model expression ability and simplify system model[1]. Another extension is introducing the time concept into the definition of Petri nets, that forms Timed Petri Nets(TPN), Stochastic Petri Nets(SPN) and generalization Stochastic Petri Nets (GSPN), Petri nets with time factors are useful in the occasion of performance analysis. Some extensions, such as Timed Colored Petri Nets(TCPN), combine above two methods in their definitions.

But Petri nets extension does not solve above problem radically, because it cannot overcome the questions met in the process of system development in the stand of software engineering, so net compression technologies are researched.

The first question of net compression is to simplify net model by transformation under certain precondition. The approximate solution of no product form solution SPNs was discussed in reference [2], where the idea of Maries method was introduced; flow equivalent nets for the performance analysis of flexible manufacturing systems was presented in reference[3], where suitable subnets were compressed and represented by corresponding transitions.

The other question in net compression is the modularity and hierarchy in net model[4], layered module method is not only the requirement of system designing and modeling, but also the demand of performance analyzing and model evaluating.

We presented the concept of engineering subnet and Engineering Petri net in reference [5], in this paper we mainly discuss the technologies of net model transformation based on engineering subnet, and we will prove that some important properties can be reserved after model transformation.

The paper is arranged in the following way: section 2 investigates the rationality to apply Petri nets in structural system design; section 3 defines the transition refinement and subnet abstraction operations; section 4 discusses the properties reserving question during subnet abstraction; section 5 discusses the properties reserving question during transition refinement; an example is given in section 6 and a few conclusions are given in section 7.
The basic idea of structural system design is function decomposition (or refinement) and system abstraction. Decomposition is used in complex systems in order to decrease complexity; usually a complex question is divided into several sub-questions which will be solved separately. The top layer model briefly describes the system across-the-board; the details are described in the bottom layers, while the middle layers reflect the transition from abstraction to detail. The decomposition operations are performed hierarchically, essential properties are taken into account in a higher layer, the details or branches are omitted at the beginning, and they will be taken into account in lower layers step by step until a completely described model is reached. The process to represent a system briefly with some essential properties is abstraction. Abstraction and decomposition operate reversely; the abstraction process is from bottom to top, while the decomposition process is from top to bottom. Figure 1 shows the two operations.

Figure 1. System abstraction and decomposition

Modularity and hierarchy play important roles in structural system design process, to realize modularity and hierarchy in Petri nets models, the concept “subnet” is crucial, usually a subnet corresponds to a module. We defined subnet in reference [6] and classified subnets into several types as figure 2, characteristics and properties of every different subnet are investigated in detail.

J. Padberg divided a complex system model into many subnets satisfying certain requirements in reference [4], he solved the problem in an algebraic way; reference [7] also researched the methods to realize subnet division and the holding of properties during the operation; in reference [8] the preservation of liveness and deadlock freeness in the synchronous synthesis of Petri net systems was researched.

A structural system designing method describes and analyzes systems hierarchically, the hierarchical model method brings at least following 3 advantages:

(1) Because the interior details are hidden through abstraction, the designer can pay attention to a whole picture and consider the macroscopically structure carefully without the interfering from a large quantity of minutiae;

(2) Module reusing can decrease workload in system designing and maintaining greatly;

(3) The structure of a model seems clear, and this helps to the abstraction and decomposition operations.
reference [9], reference [1] defined a kind of layered colored Petri nets. J. M. Proth. presented a controllable output net (CO net) in reference [10]. CO nets are reversible, live and bounded, furthermore, these Petri nets have output transitions which can be fired independently from each other. A scheme using Petri nets refinements was proposed in Reference [11], where a refined Petri net was obtained by using two kinds of subnets to replace some transitions or places in an ordinary Petri net.

III. TRANSITION SUBNET BASED ABSTRACTION AND REFINEMENT

A subnet abstraction is the process of replacing the complex subnet with a relative simple structure or component such as a transition or a place, it is the most important method of structural design in a Petri nets model, and subnet abstractions can simplify a model effectively.

According to reference [12], net abstractions are divided into 2 types as simple abstraction and strict abstraction, in this point of view, the content researched in this paper is the extension of the simple abstraction, because strict abstraction deal with not the subnets but the abstraction of other subnets including P-subnet, PT-subnets and TP-subnets operate differently, but the principles are similar.

Compared with subnet abstraction, subnet refinement operates by contraries, it is the process of replacing a transition or place with a much more detailed subnet; similar to that of subnet abstraction, the refinement in this paper is the extension of simple refinement based on transition subnet.

Definition 1: Let net \( N_1 = (P_1, T_1; F_1, M_0_1) \), given \( SN = (P_5, T_5; F_5, M_0_5) \) is a T-subnet of \( N_1 \), net \( N_2 = (P_2, T_2; F_2, M_0_2) \), \( t_y \) is a transition of \( N_2 \), if:

1. \( P_2 = P_1 \setminus P_5 \);
2. \( T_2 = (T_1 \setminus T_5) \cup \{ t_y \} \);
3. \( F_2 = \{(x, t_y) \mid \exists y \in T_5 \mid x \in P_2 \land (x, y) \in F_1 \} \cup \{(t_y, x) \mid \exists y \in T_5 \mid x \in P_2 \land (y, x) \in F_1 \} \cup \{(x, y) \mid x \in P_2 \land y \in T_5 \land (x, y) \in F_1 \} \cup \{(y, x) \mid x \in P_2 \land y \in T_5 \land (y, x) \in F_1 \} \);
4. \( \forall p \in P_2 : M_0_2(p) = M_0_1(p) \).

Then \( N_2 \) is said to be the T-abstraction of \( N_1 \) based on T-subnet \( SN \), and represented as \( N_2 = TAbst(N_1, SN) \).

Abstractions of T-subnets in figure 2 are shown in figure 3.

Definition 2: Let net \( N_1 = (P_1, T_1; F_1, M_0_1) \), given \( SN = (P_5, T_5; F_5, M_0_5) \) is a transition subnet or net, \( N_2 = (P_2, T_2; F_2, M_0_2) \), \( t_y \) is a transition in \( N_2 \), if:

1. \( P_2 = P_1 \setminus P_5 \);
2. \( T_2 = T_1 \setminus T_5 \cup \{ t_y \} \);
3. \( F_2 = F_1 \setminus \{ (x, y) \mid x \in P_2 \land (x, t_y) \in F_1 \} \cup \{(x, y) \mid x \in P_2 \land (y, x) \in F_2 \} \cup \{(y, x) \mid x \in P_2 \land (y, x) \in F_1 \} \);
4. \( \forall p \in P_2 : M_0_2(p) = M_0_1(p) \).

Then \( N_1 \) is said to be the T-refinement of \( N_2 \) about transition \( t_y \) based on subnet \( SN \), which is represented as \( N_1 = TRfine(N_2, t_y, SN) \).

Figure 3. Typical T-subnets and their abstraction

Transitions in figure 3 are refined as corresponding subnets in figure 2.

Instead of requiring subnets to be alive, we defined the concept of engineering subnet in reference [5]. In an engineering subnet the sub system will not reach a “not live” marking from the initial marking, where a “not live” marking means there exists no way to fire the output transition; an engineering subnet also makes sure that there exists no “dead” loopback, where such loopbacks will bring the system into a “not live” marking. An engineering subnet \( SN \) is a subnet with the ability to execute continuously, it owns live loopbacks and the total number all loopbacks in it is finite; such a live loopback corresponds to a processing route, a not live loopback corresponds to a initialing process, the subnet will not trap into a “not live” state.
Definition of engineering subnet takes fault tolerant into account. If a subnet is required to be alive, it means every transition in the subnet should be alive, but this limitation usually collides with practical application. A trifling trouble in a module may cause the failure of certain non key functions (corresponding transitions become not alive), but if the main functions of the module keep effective, to its surroundings, the module is still workable. The presupposition of an engineering subnet is that the principal parts of the subnet can work normally, so the subnet doesn’t affect the function of the rest system, namely its test net owns live loopbacks, i.e. the module can input and output normally.

IV. PROPERTIES RESERVING IN SUBNET ABSTRACTION

Important properties including boundedness, liveness and reversibility are expected to be reserved in model transformation. Liveness indicates no deadlock (but deadlock free does not means liveness), thus a system can run successfully; boundedness and safety guarantee that the number of tokens in every place will not exceed a certain limitation, thus a system can run in gear with no overflow; reversibility means a system run periodically, so some of the functions or operations can be repeated. Of course to reserve these properties both the subnet and the transformation process should satisfy certain requirements, in other words, some limitations should be assigned to the subnets and transformations.

If no extra explanation is given in the rest of this paper, net \( N_1 = (P_1, T_1; F_1, M_01) \), net \( N_2 = (P_2, T_2; F_2, M_02) \), and
\( SN = (P, T; F, M_0) \) is an engineering subnet for \( N_1 \), \( t_x \) is the corresponding transition of \( SN \) in the process of abstraction or refinement.

According to our previous researches in reference [5], A normalized T-subnet SN is called an Engineering Subnet if:

1. \( \forall M_{SN} \in M_0 \), if \( M_{SN}[t_1] > 0 \) then \( t_1 \) is included in a loopback \( T^2 \),

2. \( Loop(SN) = L P_1 \cup L P_2 \), where \( L P_1 \) is a non-empty, finite set of live loopbacks, \( L P_2 \) is a finite set of loopbacks; and \( \forall \tau \in L P_2 : M_{02}[\tau > M_{SN} \land M_{SN}[\tau] \in LR2(SN) \), or \( \exists \tau_1, \tau_2, ... \), \( \tau_2, ... \in L P_2 \), \( M_{02}[\tau_1, \tau_2, ... \tau_2, ... > M_{SN} \land M_{SN}[\tau] \in LR2(SN) \).

A live and normalized subnet owns and only owns live loopbacks, in a live and normalized \( T^2 \)-subnet SN, the firing of \( t_o \) is the sufficient and necessary condition to fire \( t_x \). A live and normalized T-subnet with finite loopbacks is an Engineering Subnet.

As an engineering subnet, the properties and specialties owned by \( SN \) form the basis to discuss the question of properties preserving in this paper. Existence of live loopbacks is the most important property of engineering subnets, compared with reference [10] and [11], this precondition for subnet is much more relaxed.

Subnet abstraction is the operation to simplify a system model, so usually different properties of the original model keep tenable after the transformation. The following results come into existence.

**Corollary 1:** Given \( N_2 = TAbst(N_1, SN) \), if \( N_1 \) is bounded (or safe), then \( N_2 \) is bounded (or safe).

Proof: If \( N_1 \) is bounded, then for \( \forall p \in P_1 \), there must exist a non negative integer \( l_1 \), that \( \forall M_{SN} \in R(M_0) \), \( M_{SN}(p) \leq l_1 \); for \( \forall p \in P_2 \), \( p \in P_1 \) and it belongs to one of the following cases:

1. If \( p \notin t_x^2 \), then the refinement operation makes no influence to \( p \), so \( p \) keeps bounded.

2. If \( p \notin t_x \), tokens in \( p \) are produced the same as that in \( N_1 \), \( t_x \) in \( N_2 \) consumes tokens the same way as \( t_y \) of \( SN \) in \( N_1 \), so \( p \) keeps bounded.

3. If \( p \notin t_x^* \), tokens in \( p \) are produced by \( t_x \); according to properties of engineering subnet, each firing of \( t_1 \) in \( SN \) results a firing of \( t_0 \); the case to fire \( t_x \) is the same as that to fire \( t_y \). So each firing of \( t_x \) in \( N_2 \) indicates a firing of \( t_y \) in \( N_1 \), and brings a firing of \( t_0 \); the same way, a firing of \( t_0 \) indicates a firing of \( t_x \). So under corresponding state in \( N_1 \) and \( N_2 \), \( t_x \) produces tokens for \( p \notin t_x^* \) the same way as \( SN \), since \( p \) is bounded in \( N_1 \), \( p \) keeps bounded in \( N_2 \).

With (1),(2) and (3) together, the boundedness is proved to be reserved, we can prove the reservation of safety similarly.

The following corollaries come into existence intuitively, since \( P_2 \) is only a subset of \( P_1 \), \( P_1 \in P_2 \) and \( T_2 - \{t_x\} \in T_1 \), so the behaviors and states of \( N_2 \) is only part of \( N_1 \), properties satisfied in a complete system keep effective in a part of the system of course. In fact, the subnet abstraction makes the operation and state of a system model simply.

**Corollary 2:** Given \( N_2 = TAbst(N_1, SN) \), if \( N_1 \) is live, then \( N_2 \) is live.

**Corollary 3:** Given \( N_2 = TAbst(N_1, SN) \), if \( N_1 \) is reversible, then \( N_2 \) is reversible.

**Corollary 4:** Given \( N_2 = TAbst(N_1, SN) \), if \( N_1 \) is deadlock free, then \( N_2 \) is deadlock free.

The proofs of above results are similar to that of corollary 1 and are omitted in this paper.

V. PROPERTIES RESERVING IN TRANSITION REFINEMENT

Behaviors of a system become more complex after the transition refinement operations, so certain preconditions must be satisfied to preserve properties in the original model. Prior to the discuss of refinement operation, it should make clear that the places in a net model are divided into 2 parts, namely \( M_1 = (M_2, M_3) \).

© 2010 ACADEMY PUBLISHER
**Definition 3:** Let $M_i = (M_2, M_3)$, where $M_2$ is formed by places of $N_3$, represented by $MPN(M_i)$; $M_3$ is formed by places of $N_2$, represented by $MPSN(M_i)$.

The main difference between the two nets before and after refinement is caused by the concept “half fired” we present in following definition 4.

**Definition 4:** Given $SN$ is a $T^e$ engineering subnet, $N_i = TRefine(N_2, t_N, SN)$, $\forall M_{i11} \in R(M_{01})$, if $MPN(M_{i11}) \notin R(M_{01})$, $M_{i11}$ is said to be a half fired state relative to $N_2$.

A half fired state in $N_i$ is the state when subnet $SN$ is firing, namely the input transition $t_\alpha$ has fired already but the output transition $t_\beta$ does not fire yet; in corresponding net $N_3$ the transition $t_N$ has consumed tokens from its input places but no token has been produced in its output places yet, $t_N$ is in the process of firing, such case looks similar to a transition with time delay in a time Petri net model. If $M_{i11}$ is a state where $t_\alpha$ has fired $k$ times but corresponding $t_\beta$ has fired zero times, $M_{i11}$ is called $k$ times half fired state.

In the occasion that time is not taken into account, the half fired state $M_{i11}$ has no equivalent state in $N_2$, but there exists 2 correlative states. Suppose $M_{i11}$ is a $k$ times half fired state, $M_0[R > M_{i11}]$, let $\tau = (t_{r_1}, t_{r_1}, t_{r_2}, t_{r_3}, \ldots, t_{r_2}, t_{r_2}, t_{r_3}, \ldots, t_{r_3})$, where every $t_{r_j}(j = 1, k)$ is composed by transitions in $N_2$ except $t_N$, $t_{r_j}(j = 1, k)$ is composed by transitions except $t_\alpha$ and $t_\beta$, all appearance the firing of any pair $\tau = (t_\alpha, t_\beta)$ make no difference to places outside $SN$, so the transition serial $\tau$ can be adjusted as the new expression $\tau = (t_{r_1}, t_{r_2}, \ldots, t_{r_2}, t_{r_3}, \ldots, t_{r_3}, \ldots, t_{r_3}, t_{r_3})$, let $t_{r_1} = (t_{r_1}, t_{r_1}, t_{r_2}, \ldots, t_{r_2}, t_{r_2}, t_{r_3}, \ldots, t_{r_3})$, then $M_0[R > M_{i11}]$, $M_{i12}[t_{r_2} > M_{i11}]$. According to the properties of engineering subnet, we can find $t_{r_1} = (t_{r_1}, t_{r_2}, \ldots, t_{r_2}, t_{r_3}, \ldots, t_{r_3}, t_{r_3})$, $M_{i11}[t_{r_1} > M_{i11}]$, so we can left the half fired state through $t_{r_1}$. Thus the following corollary 5 comes into existence.

**Corollary 5:** Given $N_i = TRefine(N_2, t_N, SN)$, $\forall M_{i11} \in R(M_{01})$, if $M_{i11}$ is a half fired state, there will exist $t_{s_0}$, $M_{i11}[t_{s_0} > M_{i10}]$, $M_{i10}$ is a non half fired state, namely the $k$ times half fired post state in following definition 5.

**Definition 5:** Given $M_{i11}$ is a half fired state, $M_0[R > M_{i11}]$, $M_{i12}[t_{r_2} > M_{i11}]$, $M_{i11}[t_{s_0} > M_{i10}]$, where $t_{s_0} = (t_{r_1}, t_{r_2}, t_{r_3}, \ldots, t_{r_2}, t_{r_2}, t_{r_3}, \ldots, t_{r_3}, t_{r_3})$, $s_{r_0} = (t_{s_0}, t_{r_2}, \ldots, t_{r_2}, t_{r_3}, \ldots, t_{r_3}, t_{r_3})$, $t_{r_1}(j = 1, k)$ is composed by transitions in $SN$ except $t_\alpha, t_\beta$, $t_\beta$ is composed by transition in $N_2$ except $t_N$, then:

1. $M_{i11}$ is called the previous state of the $k$ times half fired state of $M_{i11}$, represented by $pre(M_{i11})$;
2. $M_{i10}$ is called the post state of the $k$ times half fired state of $M_{i11}$, represented by $post(M_{i11})$;
3. $t_{s_0}$ is called the front of the $k$ times half fired serial of $M_{i11}$;
4. $t_{s_0}$ is called rear of the $k$ times half fired serial of $M_{i11}$.

Following Corollaries come into existence intuitively.

**Corollary 6:** $\forall M_{i11} \in R(M_{01})$:

1. $MPN(M_{i11}) \in R(M_{02})$ or $MPN(pre(M_{i11})) \in R(M_{02})$;
2. $MPSN(M_{i11}) \in R(M_{03})$ or $MPSN(pre(M_{i11})) \in R(M_{03})$;
3. $MPN(post(M_{i11})) \in R(M_{02})$, $MPSN(post(M_{i11})) \in R(M_{03})$.

**Corollary 7:** Given $N_i = TRefine(N_2, t_N, SN)$, if both $N_2$ and $SN$ are bounded, then $N_i$ is bounded.

Proof: If $N_i$ is bounded, then for $\forall p \in P$, there must exist a non negative integer $l_1$, that $\forall M_{i11} \in R(M_{01})$, $M_{i11}(p) \leq l_1$; so for $\forall p \in t_N$, this result is satisfied also, given $M(p) \leq l_1 \leq l_1$; so for $\forall M_{i11} \in R(M_{01})$, if $M_{i11}$ is a $k$ times half fired state, then $k$ is finite, given $k \leq l_2$; since $SN$ is bounded, then $\forall \in P$, there must exist a non negative integer $l_1$, that $\forall M_{i11} \in R(M_{01})$, $M_{i11}(p) \leq l_1$.

1. (1) if $M_{i11}$ is not a half fired state, namely $MPSN(M_{i11}) \in R(M_{03})$, $MPN(M_{i11}) \in R(M_{03})$ : $\forall p \in MPN(M_{i11})$, $M_{i11}(p) \leq l_1$; $\forall p \in MPSN(M_{i11})$, $M_{i11}(p) \leq l_2$; let $l = \max(l_1, l_2)$, $\forall p \in P$, $M_{i11}(p) \leq l$.

2. (2) if $M_{i11}$ is a ones times half fired state, namely $MPN(pre(M_{i11})) \in R(M_{02})$, $MPSN(M_{i11}) \in R(M_{03})$ : intuitively $MPN(M_{i11}) < MPN(pre(M_{i11}))$, so $\forall p \in MPN(M_{i11})$, $MPN(M_{i11})(p) < MPN(pre(M_{i11}))(p) \leq l_1$, namely $M_{i11}(p) \leq l_1$; $\forall p \in MPSN(M_{i11})$, $M_{i11}(p) \leq l_2$; let $l = \max(l_1, l_2)$, $\forall p \in P$, $M_{i11}(p) \leq l$.

3. (3) if $M_{i11}$ is a $k$ times half fired state, namely $MPN(pre(M_{i11})) \in R(M_{02})$, $MPSN(pre(M_{i11}))(p) \leq l_1$; suppose every $t \in T_{s_0}$ produce at most $l_1$ tokens for $\forall p \in MPSN(M_{i11})$, since $SN$ is bounded, so $l_1$ is a non negative integer, $k$ is also finite ($k \leq l_2$), so $\forall p \in MPSN(M_{i11})$, $M_{i11}(p) \leq l_1 + l_2 \times l_1$; let $l = \max(l_1, l_2 + l_1 \times l_1)$, $\forall p \in P$, $M_{i11}(p) \leq l$.
With (1), (2) and (3) together, \( \forall p \in P_1 \), \( \forall M_{11} \in R(M0_1) \), there exist a non negative integer \( l \), \( M_i(p) \leq l \), so the proposition is proved.

Safety is the special case of boundedness, but sometimes we call it 1-bounded or 1-safe.

**Corollary 8:** Given \( N_i = TRfine(N_2,t_x,SN) \), even \( N_i \) is safe and \( SN \) is safe, \( N_i \) is not certainly to be safe.

Proof: according to step (3) in above proof process of corollary 7, when \( MPSN(\text{pref}(M_{11})) \in R(M0_2) \), \( \forall p \in MPSN(M_{11}) \), \( M_{11}(p) \leq l_1 + l_2 \times l_4 \), apparently \( l_1 + l_2 \times l_4 \) usually great than 1, so the proposition is proved.

**Corollary 9:** Given \( N_i = TRfine(N_2,t_x,SN) \), if both \( N_2 \) and \( SN \) are live, then \( N_i \) is live.

Proof: since \( N_i \) is live, so \( \forall t \in T_i, \forall M_{11} \in R(M0) \), \( \exists M_{12} \in R(M_{11}) : M_{12}[t] \), then the question is changed as proving the existence of \( M_{12} \). We don’t take those half fired states into account here, because according to corollary 5, a half fired state can be fired and changed to a non half fired state, that means \( MPN(M_{11}) \in R(M0_2) \), \( MPSN(M_{11}) \in R(M0_2) \). To simplify the proof process, the format \( M_i = (M_2,M_4) \) is used here, because now \( M_2 = MPN(M_{11}) \), \( M_4 = MPSN(M_{11}) \) are effective states. Let \( M_{11} = (M_{21},M_{51}) \), \( M_{21} \in R(M0_2) \), \( M_{51} \in R(M0_3) \); since \( T_i = T_i \cup T_2 - \{ t_x \} \), so \( \forall t \in T_i, t \in T_2 - \{ t_x \} \) or \( t \in T_5 \).

(1) Given \( t \in T_2 - \{ t_x \} \). Since \( N_2 \) is live, there should exists \( t_x \), brings \( M_{21}(t_x) > M_{22}, M_{22}(t) > \); if \( t_x \) is not occurred in \( t_x \), then state of \( SN \) keeps unchanged, let \( t_1 = t_x \), then \( M_{11}(t_1) > (M_{22},M_{51}) \), let \( M_{12} = (M_{22},M_{51}) \), it’s apparently \( M_{12}[t] > \); if \( t_x \) occurs in \( t_x \), \( SN \) is live indicates there should exist a live loopback, so every firing of \( t_x \) can be replaced by a live loopback (\( t_1, t_2, \ldots \)) in \( SN \), and the influences to outside environment are completely same, so let \( M_{51}(t_1,t_2,\ldots ,t_n > M_{52} \) (test subnet \( SN \) can be used here to fire continually), if \( t_2 = (t_2,t_x,t_3,\ldots ,t_n) \), let \( t_1 = (t_1,t_x,t_2,\ldots ,t_n) \), then \( M_{12}(t_1) > (M_{22},M_{52}) \), let \( M_{12} = (M_{22},M_{52}) \), apparently \( M_{12}[t] > \).

(2) Given \( t \in T_5 \). Since \( SN \) is live, we use the test subnet \( SN \) here to replace \( SN \), then there must exist \( t_x \), where \( M_{51}(t_x) > M_{52}, M_{52}[t] > \); if \( t_x \) contains loopbacks of \( SN \), the loopbacks \( t_5, t_2, \ldots \) fires in sequence, so \( t_x = (t_5, t_2, \ldots , t_n)' \), where \( t_n' \) is the part of the last unfinished loopback (if \( t_x \) contains no loopback then \( t_x = t_x \)); the firing of every loopback corresponds to a firing of \( t_x \) in \( N_2 \), if \( t_2 = (t_2,t_x,t_3,\ldots ,t_n) \), \( M_{52}[t_2] > M_{22} \); let \( t_1 = (t_2,t_3,\ldots ,t_n)' \), then \( M_{11}(t_1) > (M_{22},M_{52}) \), let \( M_{12} = (M_{22},M_{52}) \), apparently \( M_{12}[t] > \).

With (1) and (2) together, \( M_{12} \) is sure to be existed, so the proposition is proved.

**Corollary 10:** Given \( N_i = TRfine(N_2,t_x,SN) \), if \( N_i \) is deadlock free, then \( N_i \) is deadlock free.

Proof: according to the definition of engineering subnet in reference [6], if \( SN \) is an engineering subnet, \( \forall M_{51} \in M0_3 \), \( \forall t \in T_5 \), if \( M_{51}(t) > \), then \( t_i \) is included in a loopback \( r \), this indicates in a newly arrived state we can always find a transition which can be fired, so \( SN \) is deadlock free.

According to corollary 5, a half fired state is deadlock free, because it can be transformed as a normal state, so \( \forall M_{11} \in R(M0) \) can be regarded as a non half fired state, so that \( MPN(M_{11}) \in R(M0_2) \), \( MPSN(M_{11}) \in R(M0_2) \). Since \( N_2 \) is deadlock free, there must exist \( t \in T_3 \), \( MPN(M_{11})(t) > \), so \( M_{12}[t] > \), and so under any state \( M_{11} \) there exists no deadlock, namely \( N_i \) is deadlock free. So the proposition is proved.

VI. AN EXAMPLE

An example system model is given in figure 4. We use this example to show the methods discussed above, although some of the assumptions may be different from a real system, still it makes no difference to our application.

Figure 4(a) is an nth layer module, we can suppose a part will be founded, polished and chromeplated in sequence. If \( n = 0 \) then it’s the top layer model of a complete system, otherwise it’s a \( T^n \) subnet in the nth layer.

\( t_1 \): provides raw parts. In a top layer model it’s a source transition which can be refined as a \( T^n \) subnet; in an nth layer module it’s the input transition \( t_1 \) of the \( T^n \) subnet which will take a part from outside interface each time.

\( t_2 \): moves the parts in \( p_1 \). It’s refined as figure 4(b) in detail.

\( t_3 \): polishes the parts in \( p_2 \). It’s not refined any longer in this example.

\( t_4 \): chromeplates the parts in \( p_3 \). It’s refined as figure 4(c) in detail.

\( t_5 \): moves the parts in \( p_3 \) from the (sub) system. In a top layer model it’s a sink transition which can be refined as a \( T^n \) subnet; in an nth layer module it’s the output transition \( t_5 \) of the \( T^n \) subnet which will each time send a part into outside interface.

Figure 4(b) is the detailed process of the founding operation corresponding to transition \( t_1 \) in figure 4(a). The model is a revised version of figure 5 in reference [5], the two machines take turns to process the inputting parts alternately.

\( t_1 \): input transition, takes parts from \( p_1 \) in figure 4(a).

\( t_2 \): processes the parts in place \( p_2 \).

\( t_3 \): processes the parts in place \( p_3 \).
t2O: outputs the parts in p24 to p5 in figure 4(a).

Figure 4(c) is the detailed process of the chromeplating operation corresponding to transition t2 in figure 4(a).

T41: input transition, takes parts from p4 in figure 4(a).

T42: prepares assistant material for T43 in place p43.

T43: processes the parts in place p41.

T44: clears or callbacks byproducts in place p44.

t4O: outputs the parts in p42 to p5 in figure 4(a).

It’s not difficult to verify the properties discussed in this paper using the example in figure 4.

VII. CONCLUSION

Modularization forms the basic idea of structural designing method: a system can be decomposed into independent modules which are corelative each other in a top-down manner; or aggregating different functional modules into subsystem and moreover forming the complete system in a bottom-up manner. Based on subnet technologies, system modularization and hierarchy can be realized by transition refinement or subnet abstraction in a Petri net model. Subnet normalization solves the problem of “looks similar” between subnet and transition, namely to make their interfaces to outside environment be expressed accordantly; live loopback in engineering subnet solve the problem of “perform similarly”, namely the transition and corresponding subnet should have almost the same actions. In a net model the main difference between subnet and transition is brought by the half fired states. Transform under certain precondition keeps important properties such as boundedness, safety, deadlock and reversibility in the original model, these properties are required when modeling physical systems. The research makes it possible to modeling complex system using Engineering Petri net, and provides a new method to develop system efficiently.

ACKNOWLEDGMENT

This paper is based on “Subnet Abstract and Transition Refinement in Petri Nets Model”, which is appeared in proceedings of WCSE2009.

Project supported by the Natural Science Foundation of Guangdong Province (Grant No. 8451032001001610 ), Natural Science Foundation of Guangdong Province (Grant No. 0630970 ).

REFERENCES

Zhijian Wang was born in Changsha, Hunan Province, China in 1970. He obtained his Ph.D. degree in Control Theory and Control Engineering from Central South University in 2007 in China.

He is presently an associate professor of Computer Science in Information Science School, Guangdong University of Business Studies, Guangzhou, China. He is also the senior member of China Computer Federation. His current research interests include Petri nets, software engineering, and Computer Integrated Manufacturing Systems.

Dingguo Wei was born in Hunan Province, China in 1964. He obtained his Ph.D. degree in Computer Software and Theory from Fudan University in 2003 in China.

He is presently a professor of Computer Science in Information Science School, Guangdong University of Business Studies, Guangzhou, China. He is also the senior member of China Computer Federation. His current research interests include Petri nets, software engineering, and Electrical business.

Cheng Xu was born in 1962. He received B.S. degree and M.S degree from Hunan University in P. R. China in 1983 and 1986 respectively. He received PhD degree in Mechanical Manufacture and Automation Wuhan University of Technology in P. R. China in 2006.

Currently he is professor in College of Computer and Communication, Hunan University, P. R. China. His current research interests include embedded system, network multimedia applications and control theory.