The Sixth International Conference on CSCW in Design. July 12-14, 2001, London, Ontario, Canada

Modeling Workflow within Distributed Systems

Yuhong Yan, Alex Bejan*
Flydragon Computer Consulting, NY, USA
ANCA Tech. RI, USA
flydragon960@yahoo.com

Abstract

Workflow management techniques are aiming at supporting business process across organization boundaries. Current techniques are lacking of the formalism tools to model and analyze workflows in a large scale. And current WFMSs do not have the ability to react to the response of another WFMS dynamically. Using Petri Nets as the modeling tools, we present the concept of Standard Workflow Structure and a set of standard workflow blocks are designed. We prove that modeling a workflow process in Standard Workflow Structures can guarantee the soundness of a workflow network defined by Aalst [1]. We also presented the decomposing method of a large scaled workflow network into sub-networks. From our analysis and prove, we draw a very good conclusion that the workflow network composed by self-loop connected sub-networks maintains soundness under proper structure conditions. The methods presented are not only a design language for the specification of complex workflows, but also powerful analysis techniques to verify the workflow procedures.

1. Introduction

The main purpose of a workflow management system is the support of the definition, execution, registration and control of processes. These processes can be any kind, but most stimulation comes from its promising usage in managing business processes.

Modeling business processes is the start point of the workflow management techniques. Workflow techniques promise to provide the efficient way to model the complex business processes, so as to optimize them, reuse them and control them. But the fact is though many workflow management systems have modeling tools, these tools are paid much less effort than the workflow engine itself. These modeling tools normally use icons and links to represent the activities and the sequential order between the activities (e.g. ActionWorks, IBM FlowMark, Filenet, etc)[2][3][4]. Associated with these visual tools, description languages (e.g. ActionWorks, FlowMark)[2][3] are designed to allow linguistically programming and manipulating these process instances. However, current techniques in workflow modeling have many drawbacks. First, nearly all the methods developed so far are limited on the level of graphical description. No mathematical skills are used to analyze the properties of the processes. The correctness of the modeling is based on human check and considerations. Second, the current methods are not suitable in large scaled analysis, where the processes are more complex and across organizational boundaries. Not one workflow system can define and execute theses processes. Third, the diversity of current methods prevents the interoperation among WFMS from different vendors. Although WFMC has set up a reference model [5] to provide basic conceptions and enhance interoperation among heterogeneous workflow management systems, the situation is not improved.

We consider the Petri net is a natural model for accomplishing the need of workflow modeling. The main reasons for using Petri nets for workflow modeling [1] include: Formal semantics, a workflow process specified in terms of a Petri net has a clear and precise definition, because the semantics of the classical Petri net and several enhancement (color, time, hierarchy) have been defined formally; Graphical nature, Petri nets are a graphical language and easy to learn. This feature made Petri nets to be the choice to be used in workflow definition Tools, designed in WFMC Workflow Reference Model [5]. Expressiveness, Petri nets support all the primitives needed to model a workflow process. All the routing constructs present in today’s workflow management systems can be modeled. Analysis methods, this is a great asset in favor of the use of Petri nets for workflow modeling. The analysis techniques can be used to prove properties and to calculate performance measures. Vendor independent, Petri nets are not based on a software package of a specific vendor and do not cease to exist if a new version is released.

Using Petri Nets as the modeling tools, we present the concept of Standard Workflow Structure and a set of standard workflow blocks. We prove in this paper that modeling a workflow process in Standard Workflow Structures can guarantee the soundness of a workflow network defined by Aalst (1998). We also present the method of decomposing a large scaled workflow network into sub-networks. From our analysis and prove, we draw a very good conclusion that the workflow network composed by self-loop connected sub-networks maintains soundness under proper structure conditions.

2. Workflow Modeling Based on Petri Nets: State of Art

Petri net is the promising tool of workflow modeling. Here is its definition.

Definition 1 (Petri nets) A PN is a three-tuple (P, T, F) [6], where
1. P = {p1, p2, p3, ..., pn} is a set of places
2. T = {t1, t2, t3, ..., tn} is a set of transitions
3. $F \subseteq S \times T \cup T \times S$ is a set of acres linking the places and the transitions
4. $P \cup T \neq \Phi$, $P \cap T = \Phi$

The early work in workflow modeling lays in Zisman's work in his thesis [7]. His modeling modified the usually Petri net firing rules by adding other rules to act as guards for transition firing. Li [Li, 1990] also employed a Petri net variant for representing office work. While both of these models are essentially high-level Petri nets, their control flow is specified using the basic Petri net firing rules. [8] presents the Information Control Net (ICN) model which is intended to represent control flow and data flow in office procedures. ICNs identify firing per transition by associating a token with each transition, then using semantics similar to color Petri nets. The recent work down by Aalst [1] presents the formal definition of workflow network based on the concept of Petri Net [1]:

Definition2 (WF-net) A Petri net PN= (P, T, F) is a WF-net (Workflow net) if and only if

(1) PN has two special places: i and o. Place i is a source place $\bullet i = \phi$. Place o is a sink place: $o\bullet = \varnothing$.

(2) If we add a transition t* to PN which connects place o with i (i.e. $\bullet t^* = \{o\}$ and $t^* \bullet = \{i\}$), then the resulting Petri net is strongly connected.

Aalst also discusses the correctness of the model. He considers a correctly modeled workflow should transfer the token from the start place to the end place and when there is a token at the end place, there is no other token in any other places. Second condition is that any transition in the workflow net can be triggered in a path through out the network. He gives this property a name of soundness.

Definition 3(Sound) A procedure modeled by a WF-net PN = (P, T, F) is sound if and only if

(1) For every state M reachable from state i, o is reachable. Formally:

$\forall M$, if $(i[\sigma > M))$, then $\exists f$, $M[f > o$

where σ is transition series

(2) State o is the only state reachable from state i with at least one token in place o. Formally:

$\forall M$, if $i[\sigma > M \wedge M = o$ then $M = 0$

(3) There are no dead transitions in (PN, i). Formally:

$\forall t \in T$, if $\exists M, M'$ $i[\sigma > M$, then $\exists t$ $M[t > M'$

The soundness of a WF-net implies the correctness of the model. Our following analysis based mostly the work of Aalst.

3. The Definition of Standard Workflow Network

The motivation of defining the standard workflow network comes from the observation of the illness structures a workflow net concept can conduct. See the two workflow nets show in figure 1.

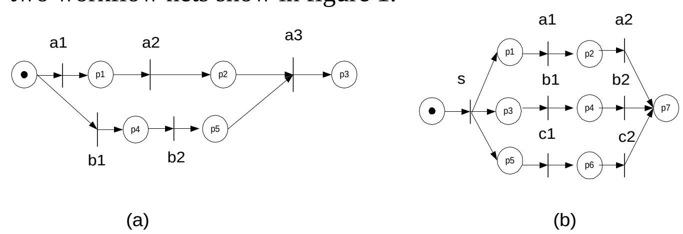

Figure 1. Two Samples of Illness Structures

Figure 1 summarizes two kinds of ill structures. In (a), following the source place are two Or-branches, so only one branch can be traced. But no matter which branch is executed, the token is blocked in p2 or p5, because the transition a3 can never be triggered. The reason of this illness comes from the structure itself. Two Or-branches after an Or-split point are synchronized by one transition. This causes this transition dead. A symmetric case is in (b), where the And-branches are synchronized by one Or-join point. This causes the problem of resource competition and this structure cannot be ended properly.
This means that the structure of a workflow network should have more restrict limitation, so that the Or-split point and And-split point cannot appear as a pair. In the following section, we introduce the definition of Standard Workflow to avoid this problem.

Definition 4 (the Basic Sequential Branch) the procedure formed by sequential linked transitions and places. Noted by Seq_Branch(P, T, I, O). Formally:

Seq_Branch(P, T, I, O)=[P0][tkPk]*,

where palces: P={p0, p1,..., pn}, and transitions T={ t0, t1,..., tn }. $\forall i=0,1,...n-1$, $I(pi,tt+1)=1$; $\forall i=1,2,..,n$, $O(pi,ti)=1$; other $I(pi,tj)=0$, $O(pi,tj)=0$. P0(t1) is called the start place (transition)of the branch, pn(tn) is called the end place (transition) of the branch. See figure 2 (a).

Definition 5 (the Basic Conditional Branch) a procedure formed by several basic sequential branches which share the common start place and the common end place. Note by Cond_Branch(P, T, I, O). Formally

$$[p_s][\bigcup_{k}^{m}\{t_{k,0}Seq\_Branch(P_k,T_k,I_k,O_k)t_{k,nk}\}][p_e]$$

ps(pe) is called the start (end) place of each branch, tk,0(tk,nk)is the start (end) transition of the kth branch. See figure 2 (b).

Definition 6 (the Basic Parallel Branch) a procedure formed by several basic sequential branches which share the common start transition and the common end transition. Note by Par_Branch(P, T, I, O), formally:

$$[p_s][t_s][\bigcup_{k}^{m}\{t_{k,0}Seq\_Branch(P_k,T_k,I_k,O_k)t_{k,nk}\}][t_e][p_e]$$

ts(te) is called the start (end) transition of each branch, pk,0(pk,nk)is the start (end) place of the kth branch. See figure 2(c).

Definition 7 (the Complex Sequential Structure) the procedure formed by serializing the basic sequential branches, the complex conditional structures and the complex parallel structures. Noted by Seq_Struct(P,T,I,O), formally:

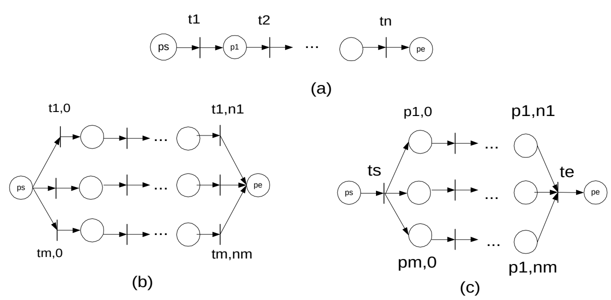

Figure 2. The basic structure of workflow network

$$\{[Seq\_Branch(P_s,T_s,I_s,O_s)]^*[Cond\_Struct(P_c,T_c,I_c,O_c]^*[Par\_Struct(P_p,T_p,I_p,O_p)]^*\}^*$$

Definition 8 (the Complex Conditional Branch) the procedure formed by paralleling several complex sequential structures with the common start place and the end place. Noted by Cond_Struct(P,T,I,O), formally:

$$[p_s][\bigcup_{s}^{m}\{t_{s,0}Seq\_Struct(P_s,T_s,I_s,O_s)t_{s,ns}\}][p_e]$$

Definition 9 (the Complex Parallel Branch) the procedure formed by paralleling several complex sequential structures with the common start transition and the common end transition. Noted by Par_Struct(P,T,I,O), formally:

$$[p_s][t_s][\bigcup_{s}^{m}\{t_{s,0}Seq\_Struct(P_s,T_s,I_s,O_s)t_{s,ns}\}][t_s][p_e]$$

Definition 10 (the Standard Workflow Network (SWF-net)) the network formed by the above complex structures with shared start place and end place.

As a stronger definition than the WF-net, SWF-net inherits the properties of the WF-net (i.e. the common start place and the end place, the strong connection). In real work, we found its logic covers all kinds of the business processes, so that it can be used in most applications. The SWF-net benefits at least too things. First, it can be used in designing workflow engine. The execution algorithms are designed according to these standard blocks. Second, the SWF-net can be used in large scaled system analysis. As we will discuss in section 3.2 the large scaled network can be divided into several sub-networks while maintaining its properties, which also found the theoretical base to link several distributed networks together.

A SWF-net has the following properties:

Theorem 1. A SWF-net (N, i) is bounded.
Theorem 2. A SWF-net (N, i) is live
Theorem 3. A SWF-net is sound.

According to the definition of SWF-net, the first two theorems are easy to prove. The third theorem is based on the Theorem 1 in [1]. The three theorems provide the fundamental for workflow decomposition.

4. Workflow Network Decomposition

4.1 Workflow Network Decomposition

A large scaled workflow process normally involves several companies or several departments in one company. Thus a workflow network is composed of several sub-networks that are modeling independently and executed by several separated workflow management systems. A sample process of order processing is shown in figure 7. The specific transitions and places are listed in table 1 in Appendix.

The sales department (CA) accepts orders from customers, and then passes the order to production department (SA). The production department therefore makes the production plan. In order to gather enough information about inventory, the production department must check the inventory management department (PMA). Thus SA responses to CA after PMA sends it answers. And CA responses to customer request according to SA's report. Figure 3 is the whole Petri Net model.

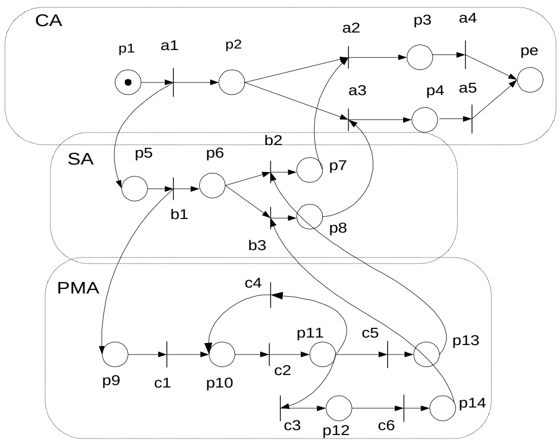

Figure 3 order processing

In the real work, the whole process is managed by three departments. This makes us to consider how to divide the whole network into three sub-networks. To decompose a network, we must consider three problems: 1) how to divide the whole network, 2) how to build the synchronous relation between sub-networks, 3) the properties of the sub-networks.

For the first problem, one nature division is to divide the network by nature boundaries of organizations, for normally organization will have an independent workflow engine. For the case in figure 3, the network can be divided into three sub-networks and each sub-network corresponds to the tasks of each department. In order to make sub-network compliant to the definition of SWF-net, a start place and an end place normally need to be added into the sub-networks. For the case in figure 3, the sub-network of SA and PMA must be added with a start place and an end place respectively.

The adding of the new start places and end places is reasonable, for sub-networks are controlled by different WFMSs and each should have starting and ending conditions. Since the starting conditions are not only related to the state the local sub-network, but also related to the other sub-networks, the definition of the added start places and the added end places are different.

The second problem is the synchronization of the sub-networks. From figure 3, man can see the CA controls the start of the SA at the transition a1, while SA controls which branch of the two Or-connected branches in CA. SA and PMA have similar relations.

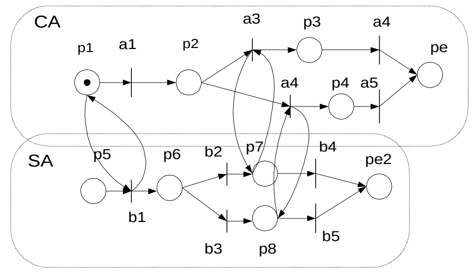

Figure 4. Sub-network Decomposition

Taking out CA and SA from figure 3, we get two sub-networks in figure 4. Note that the new start place and the new end place are added into figure 3. The arc (p1,b1), (b1,p1) form a loop. This loop has no influence on CA, while SA is controlled by CA. Similarly, (a3, p7)(p7, a3) and (a4, p7)(p7, a4) form two other loops, but SA controls CA. This loop is controlled as self-loop in Petri net [9].

Definition 10 (Self-loop) two sub-network N1=(P1,T1,I1,O1), N2=(P2,T2,I2,O2), $T1\cap T2=\varnothing$, $P1\cap P2=\varnothing$, and two initial state is M1, M2. If $\exists p^*\in P1$, $t^*\in T2$, $I(p^*,t^*)=O(p^*,t^*)=1$, a self loop is formed from N1 to N2. p* is called connection place in this self loop, t* is called connection transition in this self loop. N1 is the active network and N2 is the passive network. If whole network is noted by N=(P,T,I,O),then $P=P1\cup P2$, $T=T1\cup T2$, $I= I1\cup I2\cup I(p^*,t^*)$, $O= O1\cup O2\cup O(p^*,t^*)$, M=[M1, M2].

Until now, we get one conclusion:

A standard workflow can be divided into several self-loop connected sub-networks.

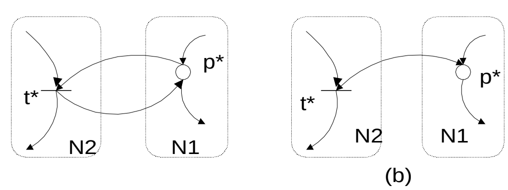

Figure 5. Self-loop connection

In real work, we are facing a reversed process. A business process across organization bounded normally is modeled and managed by several independent WFMSs and actually none knows the whole picture of the entire process. Thus, we are facing the third problem—if each WFMS models its work in the proper way (i.e. guarantee the soundness of its local network), how are the properties of the network made up of the sub-networks?

4.2 The property maintenance of sub-networks

In this section, we discuss the conditions of property maintenance of sub-networks. More actually, the question is if the sub-networks are sound, under what condition can the joint network keep the property? After several lemmas, we present the sufficient and necessary conditions for property maintenance. Some of the following work has link to [10], but our work is based on definitions for workflow network.

Lemma 1. Suppose N1 and N2 are two SWF-nets, N1 is the active net and N2 is the passive net. N1 and N2 make up a joint network N, note the reachable set under state M is R(N,M), then for $\forall$M*=[M1*, M2*]$\in$R(N,M),M1*$\in$R(N1,M1), M2*$\in$R(N2, M2).

Prove: see appendix.

Lemma 2. Suppose N1 and N2 are two SWF-nets, N1 is the active net and N2 is the passive net. N1 and N2 make up a joint network N, then (1) N is bounded; (2) the necessary and sufficient condition of N1 and N2 and N are live.

Prove: see appendix.

Lemma 3. Suppose N1 and N2 are two SWF-nets, N1 and N2 make up a joint network N. There are two self-loops. One of the self-loops is from N1 to N2 , and the other is from N2 to N1. Then (1) N is bounded; (2) the joint workflow N is live iff (a) if pi* and ti* are on the same sequential branch, then $\forall$i (i=1,2), pi* is the preset of ti*; (b) if ti* and pi* are not on the same branch, then $\forall$i (i=1,2) ti* is at the start transition of the conditional structure; or (c) if ti* is on a branch of a parallel structure, pi* is the preset of the end transition of this structure.

Prove: see appendix.

We build the extended network of the joint workflow network in figure 6 in order to prove the condition of sound for a joint workflow network.

Theorem 4. If the two standard workflow network N1 and N2 are sound, then their joint workflow network N keeps sound under the conditions of Lemma 3.

Prove: removed for context limitation.

Theorem 4 is the most important result of this paper. If we satisfy the three conditions in Lemma 3, we can safely link the multiple workflow networks and assert the joint workflow is sound.

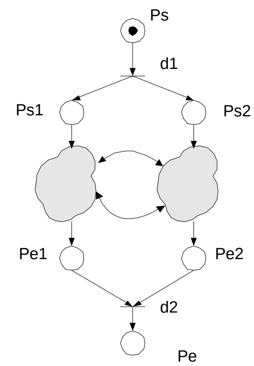

Figure 6. Joint Workflow

It is very useful in real work. Normally we can guarantee a single workflow network is sound. With the above theoretical analysis, the workflow network can keep sound if certain conditions are satisfied. But we do not emphasize the three conditions must be satisfied, because there is no picture of the whole network. The meaning of theorem 4 is not only on providing the necessary and sufficient conditions, but also on providing the criteria when the exception handling activities should be added.

5. Conclusion

This paper presents a set of methods for modeling large-scaled workflow network. Using the concept of Standard Workflow Net presented in this paper, the soundness of single workflow net and the joint workflow net can be guaranteed. The theorem 2 can be inferred to multiple self-loop cases or multiple sub networks (>2). This discuss is not done in this paper.

This method concerns only the determined relationship between activities. It is not suitable in analyzing the un-determined relationships, e.g. several activities can be executed in un-determined sequence. For some more complex situations, high Petri Nets (colored, timed) might help. But their verification is much more difficult. Another reason for not using high Petri Nets is that those complex methods are unlikely to be used in commercial products for training cost is too high and unfeasible. If the user can change the description of the problem a little bit and concentrate only to the final goal, the classic Petri Nets would be able to describe any relations, at least not worse than the block diagram provided by WfMC.

Self-loop connection might be only one control relation between sub networks. We are going to investigate other control relations.

References:

[1] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, 8(1), p21-66, 1998.
[2] Action Technology Inc. "ActionWorks Developer's Guide Version 5", *White paper*, 1999.
[3] IBM, "MQSeries Workflow", *White Paper*, 2000.
[4] FileNet, "FileNET's eProcess Offering: The Future of Web-Based Business Process Management", *White Paper*, 2000.
[5] WFMC, "The Workflow Management Coalition Specification, Workflow Reference Model", *Document Number TC00-1003*, Nov. 1994.
[6] Zhou, M. C. and DiCesare, F., *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Boston: Kluwer Academic Publishers, 1993.
[7] ZISMAN, M. D., "Representation, specification and automation of office procedures", *Ph.D. dissertation*, Univ. of Pennsylvania, Wharton School of Business, Philadelphia, Pa. 1977.
[8] Ellis, Clarence and Nutt, Gary, "Modeling and Enactment of Workflow Systems", *Advances in Petri Nets*, pp. 1-16, June 1993, Springer Verlag.
[9] Ferrarini, L., Narduzzi, M. and Tassan-Solet, M., " A new Approach to Modular Liveness Analysis Conceived for Large Logic Controllers' Design", *IEEE Transactions on Robotics and Automation*, Vol. 10, No.2, p169-184, April,1994
[10] Zhibin Xu, "Performance Analysis of Petri Net for a class of DEDS", Master Thesis, Tsinghua University, China, 1995.

Appendix:

1. Table.1 the definition of places and transitions in figure 2.

Transactions	Description	Places	Description
a1	send new order to SA	p1	start
a2	get Commit message	p2	wait
a3	get Reject message	p3	success
a4	report Commit	p4	fail
a5	report Fail	pe	end
b1	send new order to PMA	p5	get_order
b2	send Commit message to CA	p6	wait
b3	send Reject message to CA	p7	success
c1	divide order into parts	p8	fail
c2	send production schedule to work unit	p9	get_order
c3	get Reject message from work unit	p10	planned
c4	get Commit message from work unit	p11	wait
c5	report Commit to SA	p12	get_refuse
c6	send Reject message to SA	p13	success
		p14	fail

2. Prove of Lemma1:

Lemma 1. Suppose N1 and N2 are two SWF-nets, N1 is the active net and N2 is the passive net. N1 and N2 make up a joint network N, note the reachable set under state M R(N,M), then for $\forall M^*=[M1^*, M2^*]\in R(N,M)$, $M1^*\in R(N1,M1)$, $M2^*\in R(N2, M2)$

Prove: (1) from the definition of reachable set, has $\forall M^*=[M1^*, M2^*]\in R(N,M)$, $\exists f$, $M[f>M^*$. Consider different cases of t. Before t happens, $M^*=[M1^*, M2^*]$, $M1^*\in R[N1, M1]$, $M2^*\in R[N2, M2]$; after t happens $M''=[M1'', M2'']$, $M''(p)=M^*(p)+O(p,t)-I(p,t)$. Suppose T^* is the self-loop transition set and $T^*\subseteq T2$, P^* is the self-loop place set and $P^*\subseteq P1$, M1', M2' are the marks when t happens in each individual net without considering the link of self-loop.

$t\in T1$:

when $p\in P1$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M1^*(p)+O(p,t)-I(p,t)=M1'(p)$;

when $p\in P2$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M2^*(p)+0-0=M2^*$

$t\in T2\backslash T^*$

when $p\in P1$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M1^*(p)+0-0=M1^*(p)$;

when $p\in P2$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M2^*(p)+O(p,t)-I(p,t)=M2'(p)$;

$t\in T^*$

when $p\in P1\backslash P^*$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M1^*(p)+0-0=M1^*(p)$;

when $p\in P^*$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M1^*(p)+O(p,t)-I(p,t)=M1^*(p)$;

when $p \in P2$, $M''(p)=M^*(p)+O(p,t)-I(p,t)=M2^*(p)+O(p,t)-I(p,t)=M2^*(p)$;

From above, after t happens, $M1'' \in R(N1,M1)$, $M2'' \in R(N2,M2)$. From the initial mark $M=[M1, M2]$, let $f=t1t2...tn$, then $M[f>M^*=M[t1> M' [t2> M''[...[tn>M^*$, it can be deduced $M1' \in R[N1, M1]$, $M2' \in R[N2, M2]$, $M1'' \in R[N1, M1]$, $M2'' \in R[N2, M2]$, ..., $M1^* \in R[N1, M1]$, $M2^* \in R[N2, M2]$, so we can get $M^* \in R(N1,M1) \times R(N2,M2)$.

(2)if transition serial σ has only transitions in T1, from (1), get $[M1, M2][\sigma>[M1^*,M2]$, then for the joint network $M=[M1^*, M2]$, $M1^*$ is sub-vector of M.

(3) as (2), if σ has only transitions in T2, then proved. If $t^* \in \sigma$, then because N1 is live, t^* can be triggered, after it is triggered, $[M1, M2][\sigma>[M1,M2^*]$. So the final result is proved.

3. Prove of Lemma 2.

Lemma 2. Suppose N1 and N2 are two SWF-nets, N1 is the active net and N2 is the passive net. N1 and N2 make up a joint network N, then (1) N is bounded; (2) the necessary and sufficient condition of N1 and N2 and N are live.

Prove: (1) From lemma 1,$\forall$ $M^*=[M1^*, M2^*]$, $M1^* \in R(N1,M1)$, $M2^* \in R(N2,M2)$. And N1 and N2 is bounded, $M1^*<C1$, $M2^*<C2$, then $M^*=[M1^*, M2^*]<[C1,C2]$, so N is bounded.

(2) prove sufficient condition first, if $t \in T1$, because N1 is live, $\forall t \in T1$, $\exists M1''$, $M1' \in R[N1, M1]$, $i1[\sigma>M1''$, $M[t> M1'$. For the joint network, $M=[M1,M2]$, f is a transition serial made up of transitions in N1, then $[M1,M2][f>[M1'', M2][t>[M1', M2]$, i.e. t is live.

When $t \in T^*$, suppose from N1 to N2, there is k self-loops. Because N1 is live, $M1(pi^*) \in R[N1, M1]$, i.e. $\exists fi$, $M1(pi^*)=1$; to joint network $M=[M1,M2]$, $\exists \sigma 1$ has only transitions in T1, $fi \in \sigma 1$, $[M1,M2][\ \sigma 1>[\ M1'\ (pi^*), M2]$; to N2, because N2 is live, i.e.$\forall t \in T2$, $\exists M2''$, $M2' \in R[N2, M2]$, $i2[\sigma 2>M2''$, $M2''[t>M2'$, so $\exists M2^{\sim}$, $M2^{\sim} \in R(N2, M2')$ $ti^* \in EN(N2, M2^{\sim})$. Thus for $M=[M1,M2]$, $[M1,M2][\ \sigma 1>[\ M1'\ (pi^*), M2][\ \sigma 2>[\ M1'\ (pi^*), M2''], \exists M2^{\sim}$, $[M1'\ (pi^*), M2^{\sim}] \in R(N, [M1'\ (pi^*), M2''])$, $ti^* \in EN(N,[M1'(pi^*), M2^{\sim}])$, i.e. ti^* is live. If $t \in T2\backslash T^*$, consider only N2, there are two cases, if $\exists \sigma$, $ti^* \notin \sigma$, and $M2[\sigma>M2'$, $t \in EN(N, M2']$, then proved. If $\exists \sigma$, $ti^* \in \sigma$, and $M2[\sigma>M2'$, $t \in EN(N, M2']$, split σ to sections, $\sigma=\sigma 1 ti^* \sigma 2$, $M2[\sigma 1>M2^{\sim}$, because t^* is live, $\exists \lambda$, λ is made up of transitions in T1 and $M[\lambda>[M1^{\sim}, M2]$, then $[M1, M2][\ \lambda>[\ M1^{\sim}, M2][\sigma 1>[\ M1^{\sim}, M2^{\sim}][ti^* \sigma 2>[\ M1^{\sim}, M2'], t \in EN(N, [\ M1^{\sim}, M2'])$, proved.

4. Prove of Lemma 3

Lemma 3. Suppose N1 and N2 are two SWF-nets, N1 and N2 make up a joint network N. There are two self-loops. One of the self-loops is from N1 to N2 , and the other is from N2 to N1. Then (1) N is bounded; (2) the joint workflow N is live iff (a) if pi^* and ti^* are on the same sequential branch, then $\forall i$ $(i=1,2)$,pi^* is the preset of ti^*; (b) if ti^* and pi^* are not on the same branch, then $\forall i$ $(i=1,2)$ ti^* is at the start transition of the conditional structure; or (c) if ti^* is on a branch of a parallel structure, pi^* is the preset of the end transition of this structure.

Prove: (1) From lemma 1,$\forall$ $M^*=[M1^*, M2^*]$, $M1^* \in R(N1,M1)$, $M2^* \in R(N2,M2)$. And N1 and N2 is bounded, $M1^*<C1$, $M2^*<C2$, then $M^*=[M1^*, M2^*]<[C1,C2]$, so N is bounded.

(2) sufficient condition: when condition (a) is satisfied, if $t=t1^*$, because N1 is live, then $\exists \sigma 1$, $t \notin \sigma 1$, $M1^*[\sigma 1>M1'$, $t \in EN(N1,M1')$. Because N2 is live, then $\exists \sigma 2, t \notin \sigma 2$, $M2^*[\sigma 2>M2'$, $M2'(p2^*)>0$. Now if $t2^* \notin \sigma 2$, then $M^*[\sigma 1 \sigma 2>M'$, $t \in EN(N,M')$. If $t2^* \in \sigma 2$, note $\sigma 2=f1t2^*f2$, from $t1^*$ is enabled, and $pi^* \in PRE(ti^*)$, get $M1'(p1^*)>0$, $M1'[t2^*>M1'$, then $[M1^*,M2^*][\sigma 1> [M1' ,M2^*][\ f1\ t2^*> [M1' , M2^{*\prime}]$ $[f2 > [M1' ,M2']$, $t \in EN(N,M')$. Similarly, it can be proved when $t=t2^*$. If $t \in T1\backslash t1^*$, because N1 is live, if $\exists \sigma 1$, $t1^* \notin \sigma 1$, $M1^*[\sigma 1>M1'$, $t \in EN(N1,M1')$,then proved. If $t1^* \in \sigma 1$, $\sigma 1=g1t1^*g2$, because N2 is live, get $\exists \sigma 2$, $t \notin \sigma 2$, $M2^*[\sigma 2>M2'$, $M2'(p2^*)>0$. Now if $t2^* \notin \sigma 2$, then $M^*[\ g1t1^*\sigma 2\ g2>M'$, $t \in EN(N,M')$. If $t2^* \in \sigma 2$, note $\sigma 2=f1t2^*f2$, because $t1^*$ is enabled and $pi^* \in PRE(ti^*)$, get $M1'(p1^*)>0$, $M1'[t2^*>M1'$, then $M^*[\ g1t1^*\ f1t2^*f2g2>M'$, $t \in EN(N,M')$. Similarly it can be proved when $t \in T2\backslash t2^*$.

When condition (b) is satisfied, suppose $t1^*$ is the start transition of the branch L1. L1 is in the condition structure of N1. Because N1 is live, token can reach every place which are not on L1, including $p1^*$, then it can be deduced that transition $t2^*$ in N2 is not controlled by $t1^*$. So all transitions in N are live.

When condition (c) are satisfied, from $pi^* \notin Li$, we know branch pi^* has a token making $M(pi^*)>0$, then ti^* is enabled. Thus the end place of the parallel structure can be enabled. So all the transitions in N are live.

Necessary conditions: suppose (a)(b)(c) are not satisfied, then (1) if on a sequential branch, suppose pi^* is behind ti^*, then $ti^* \in EN(N, M(pj^*))$, $tj^* \in EN(N, M(pi^*))$, but to reach $M(pj^*)$, ti^* must be triggered, this causes a dead-lock. In another case, suppose pi^* is before ti^*, but

$pi^* \notin PRE(ti^*)$,suppose PRE(t1*)=M(p1'), we get to know, the condition to trigger t1* is that M(p1')>0 and M(p 2*)>0. Similarly, the condition to trigger t2* is that M(p2')>0, M(p 1*)>0,so when t 1* is enabled, t2* cannot be triggered, so N cannot be live.

(2) suppose pi*, ti* are on the same branch of a parallel structure, it is like condition (a), which is proved. Suppose pi*, ti* are not on the same branch of a parallel structure, and ti* is not the start place of the branch, then a transition series exists, which transfers the token to the precedent place of ti*. At this moment, M(pi*)=0, dead lock appears. (3) suppose pi* and ti* on the same branch of a parallel structure, this is like condition (a), and is proved. Suppose pi*, ti* are not on the same branch, and pi* is not in the present of the end transition of this parallel structure. Then we can construct a transition series fi, which transfers the token on the branch where pi* is on to the end place of this branch. At this moment, if ti* is enabled, M(pi*)>0 must be satisfied. And because pi* is not the end place, so M(pi*)=0. Thus if pi* is not the end place, then N is not live.