

Modeling Workflow Systems Constrained by Inputs and Outputs – An Approach Based on Petri Nets

Changyou Zheng, Yi Yao, Song Huang, Zhengping Ren

Institute of Command Information Systems, PLA University of Science and Technology, Nanjing, China

*Emails: zheng_chy@163.com
zhengpingren@gmail.com*

yaoyi226@aliyun.com

hs0317@163.com

Abstract: *Workflow systems are widely used in our daily life so that the validity, dependability and security with which they need to be assured are important. However, existing researches mainly focus on correctness validation, performance analysis and assignment scheduling, but the testing methods have been seldom suited. In this paper a formalized definition of workflows constrained by an input and output is presented, and based on that, a Petri Net-based model (I/O_WF_Net) is proposed. In I/O_WF_Net, the activities of the workflow can be modeled as transitions of a Petri Net, and the inputs and outputs of an activity can be modeled as places. After the modeling method for I/O constrained workflow net into the I/O_WF_Net model is described, the corresponding transforming algorithm and its simplifying method are given.*

Keywords: *Workflow modeling, workflow testing, Petri Net, I/O_WF_Net.*

1. Introduction

The workflow management system has been widely used in Office Automation (OA), e-Commerce, flexible manufacturing, product development and other areas. The validity, reliability and security of a workflow management system are becoming more and more important. Testing is the most reliable method for software quality assurance and it is the most common way to check whether a workflow system can run steadily. Herein, we have to say, that testing in this paper means to test the process instance while a workflow-based system is running, and it is not testing of the workflow engines. At present, the usage cases for workflow

testing are most produced by testers based on their experience, and these use cases are executed mostly manually. This has brought several weaknesses: At first, it degraded the maturity and sufficiency of testing; second, the system performance cannot be assured. Thus, there is a crying need for in-depth and pertinence study in workflow testing.

Petri net as an efficient process modeling technology that has been widely used in protocol engineering, hardware design, business process design and other fields. There are at least three good reasons for using Petri nets for workflow modeling and analysis [1].

1) Petri net is a graphical language and its semantics have been defined formally.

2) Petri net is state based instead of event based, so the state of the case can be modeled explicitly in a Petri net.

3) Petri nets are characterized by the availability of many analysis techniques.

Based on the virtues above described, a lot of model analysis and verification techniques and business scheduling methods have been developed [2-11]. However, almost all the researches on Petri net are focused on the fields above mentioned, but hardly any of them is testing oriented. In this paper we present a modeling approach based on a kind of I/O_WF_Net. In this model the activities in a workflow is defined as transitions in a Petri net, and the inputs and outputs as places. After the components and structures are modeled, an algorithm for transferring the workflow constrained by inputs and outputs to I/O_WF_Net is presented, the reduction method for I/O_WF_Net is also described.

This paper is organized as follows. Section 2 presents a review of the related work in this field. Section 3 offers a formal definition for the workflow constrained by inputs and out puts. Section 4 introduces the R/NT_WF_Net model and the modeling approach for a workflow constrained by inputs and out puts. An algorithm for transferring the workflow constrained by inputs and outputs to I/O_WF_Net and a set of reduction rules for R/NT_WF_Net are also proposed in Section 4. Section 5 concludes the paper and gives some future research.

2. Related work

The key point of workflow testing is to model a workflow. Workflow modeling has been studied for years, and by now, there has been much research in this field [12-15]. However, because of the complexity of a workflow system, it is usually difficult to present a uniform modeling method for all kinds of workflows.

As workflow testing is a new topic, there is not much research in this field. In [16] a framework for automatic dynamic testing of workflow management systems is proposed. As the syntax definitions are written in Backus-Naur Form and the whole testing script is written in XML form, the testing script is too complicated for the user to use, and the semantic is hard to understand. [17] gives an abstract model to test the web applications which use development frameworks based on the MVC design pattern and workflow paradigm. Since they only described possible strategies for testing MVWf web applications, a detailed method to model the

applications is not presented, and a detailed testing technique is not given. Quan, Lin and Wang [18] developed an automatic and scalable testing tool to evaluate the workflow systems' performance. Bartz [19] gave an automotive test data analysis method based on Petri Nets and stored by ASAM ODS.

3. Workflow constrained by inputs and outputs

Definition 1. A workflow constrained by inputs and outputs (which can be as short as I/O constrained workflow) can be defined as a six-tuple $\langle Activity, Input, Output, Relation, f_{AI}, f_{AO} \rangle$, where

1) $Activity = \{activity_1, activity_2, \dots, activity_k\} (k \geq 1)$ is the activity set of a workflow.

2) $Input = \{input_1, input_2, \dots, input_m\} (m \geq 1)$ is the input set of a workflow.

3) $Output = \{output_1, output_2, \dots, output_n\} (n \geq 1)$ is the output set of a workflow.

4) $Relation \subseteq (Activity \times Activity, Type)$ denotes the relation set of a workflow, where $Type \subseteq \{sequence, and-join, or-join, and-split, or-split\}$ is the relation type between activities.

5) $f_{AI} : Activity \rightarrow \rho(Input)$ is the input function of a workflow where $\rho(Input)$ is the power set of inputs.

6) $f_{AO} : Activity \rightarrow \rho(Output)$ is the output function of a workflow where $\rho(Output)$ is the power set of outputs.

Definition 1 presents a formal definition of I/O constrained workflow, from which we can see that:

1) The set $Activity$ includes all the activities in the workflow.

2) The set $Input$ includes all the input elements in the workflow.

3) The set $Output$ includes all the output elements in the workflow.

4) The set $Relation$ defines the relations between activities and their types. $\forall activity_1, activity_2 \in Activity$, if $(activity_1, activity_2) \in Relation$, then $activity_2$ cannot started until $activity_1$ is finished, and $activity_1$ is called the pre-activity of $activity_2$, $activity_2$ is called the post-activity of $activity_1$. There are five kinds of types between two activities: A *sequence* indicates a one-to-one relationship, which means that the pre-activity has only one post-activity and the post-activity has only one pre-activity; *and-join* indicates many-to-one relationships, which means that more than one pre-activity has the same post-activity and the post-activity cannot be executed until all the pre-activities are finished; *or-join* also indicates a many-to-one relationship, while the post-activity can execute immediately after one (or some) of the pre-activities is finished; *and-split* indicates a one-to-many relationships, which means that more than one post-activities have the same pre-activity and all the post-activities will be started after the pre-activity is finished; *or-split* also indicates a one-to-many relationship, while

one (or some) of the post-activities will be selected to be executed after the pre-activity is finished.

5) For $activity_1 \in Activity$, $input_1 \subseteq Input$, if $f_{AI}(activity_1) = input_1$ then it means that the execution of $activity_1$ needs an $input_1$; if $f_{AI}(activity_1) = \emptyset$ then the execution of $activity_1$ does not need any input.

(6) For $activity_1 \in Activity$, $output_1 \subseteq Output$, if $f_{AO}(activity_1) = Output_1$ then the implementation of $activity_1$ will output $output_1$; if $f_{AO}(activity_1) = \emptyset$ then the implementation of the $activity_1$ will not output anything.

Table I presents an example of I/O constrained workflow. The workflow is composed of 9 activities, the second and third column of the table are the inputs and outputs of the workflow, the pre-activities for each activity and the relation types are presented in the fourth and fifth column respectively.

Table 1. A workflow constrained by inputs and outputs

Activity	Inputs	Outputs	Preactivities	Relation type
A_1	p_1	p_2	\emptyset	\emptyset
A_2	p_2	p_3	A_1	<i>and-split</i>
A_3	p_2	p_4	A_1	<i>and-split</i>
A_4	p_5	p_8	A_2	<i>or-split</i>
A_5	p_5	p_9	A_2	<i>or-split</i>
A_6	p_6	p_{10}	A_3	<i>and-split</i>
A_7	p_7	p_{10}	A_3	<i>and-split</i>
A_8	p_{11}	p_{13}	A_4, A_5	<i>or-join</i>
A_9	p_{12}	p_{14}	A_6, A_7	<i>and-join</i>

4. A modeling approach for I/O constrained workflow based on Petri nets

4.1. Basic concepts of Petri nets

A detailed description for Petri net can be found in [20-22], here we only present some essential terminologies and notations used in this paper.

Definition 2. A three-tuple $N = (P, T, F)$ is named a Net if P and T are finite sets of places and transitions respectively and the following conditions are satisfied:

- 1) $P \cap T = \emptyset$, $P \cup T \neq \emptyset$,
- 2) $F \subseteq (P \times T) \cup (T \times P)$,
- 3) $\text{Dom}(F) \cup \text{Con}(F) = P \cup T$.

Some notations are presented as follows:

- 1) $t \in T$ is enabled iff $\forall p \in \bullet t$, $M(p) > 0$, and that is written as $M[t >$;
- 2) $\forall M_1, M_2, M_1 \leq M_2$ iff for all $p \in P$: $M_1(p) \leq M_2(p)$;
- 3) (PN, M_0) is live iff $\forall t \in T$, $\forall M \in R(M_0)$, $\exists M' \in R(M)$, $M'[t >$.

Definition 3. A Petri net is strongly connected iff for each point $x, y \in P \cup T$, there is a path from x to y .

Definition 4. A Petri net $PN = (P, T, F, i)$ is a Workflow net (WF_net) iff the following is satisfied:

There are special places i and o , i is a start place, i.e., $\bullet i = \emptyset$; o is an end place, i.e., $o \bullet = \emptyset$.

If a new transition t is added to PN, such that $\bullet t = \{o\}, t \bullet = \{i\}$, then the new Petri net PN is strongly connected.

4.2. I/O constrained workflow based on a Petri net

Before introducing the modeling method, a formal definition for the I/O constrained workflow based on Petri net (I/O_WF_NET) is presented as follows:

Definition 5. $\Sigma = (P, T, F, M_0)$ is a I/O_WF_NET iff the following conditions are satisfied:

- 1) (P, T, F, M_0) is a Petri net;
- 2) T represents the activity set of a workflow;
- 3) $P = P_{in} \cup P_{out}$, $P_{in} \cap P_{out} = \emptyset$, where P_{in} represents the input places and P_{out} represents the output places;
- 4) $\forall t \in T$, $p_{in} \in P_{in}$, the execution of t needs input p_{in} iff $p_{in} \subseteq \bullet t$.
- 5) $\forall t \in T$, $p_{out} \in P_{out}$, the execution of t will output p_{out} iff $p_{out} \subseteq t \bullet$.

A testing orient modeling approach for I/O constrained workflows based on Petri nets

A workflow modeling approach based on I/O_WF_NET is presented in this section. It is assumed that in a workflow, every activity needs only one input and will produce one output.

4.2.1. I/O_WF_NET model for single activity

Since a workflow is composed of many activities, so a single activity should be modeled first.

a) Single activity with an input and output

A single activity with input and output is the most common thing in a workflow. In this case, the activity is presented as a tow place and one transition, as shown in Fig. 1a.

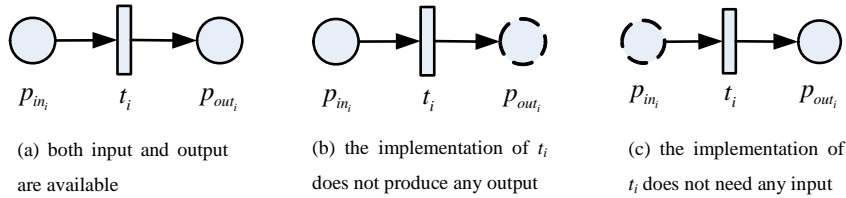


Fig 1. I/O_WF_NET model for single activity

In Fig. 1, t_i represents activity i , P_{in_i}, P_{out_i} are the input and output of t_i . If P_{in_i} contains tokens, it means that the input of t_i is satisfied and the activity can be

implemented; if P_{out_i} contains tokens, then it means that the implementation of t_i is finished, and it produced P_{out_i} . Formally, activity t_i can be denoted by $[P_{in_i}, t_i, P_{out_i}]$.

b) The implementation of a single activity does not produce any outputs

If the implementation of t_i does not produce any output, then a virtual place P_{out_i} will be added to the model. It does not mean anything. As shown in Fig. 1b, it is represented as a circle by a broken line.

c) The implementation of a single activity does not need any inputs

If the implementation of t_i does not need any input, then a virtual place P_{in_i} will be added to the model. It does not mean anything, too. As in Fig. 1c, it is also represented as a circle by a broken line.

4.2.2. I/O_WF_NET model for relationships

If $(t_i, t_j) \in Relation$, e.g., activity $[P_{in_j}, t_j, P_{out_j}]$ is the post-activity of activity $[P_{in_i}, t_i, P_{out_i}]$, then the model will depend on the relation type of the two activities.

a) The type is *sequence*

If the relation type between activity t_i and t_j is a sequence, then a virtual transition t_{ij} is added between them, such that $\bullet t_{ij} = P_{out_i}$ and $t_{ij} \bullet = P_{in_j}$, as in Fig 2. t_{ij} does not have any real semantics, it is just a bridge which connects t_i and t_j .

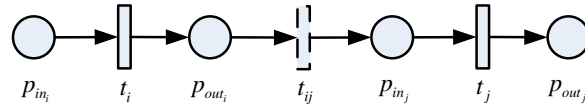


Fig. 2. I/O_WF_NET model for sequence relation

b) The type is *and-join*

If the relation type between activity t_i and t_j is and-join, then the model built here depends on whether $\bullet P_{in_j} = \emptyset$ or not. If $\bullet P_{in_j} = \emptyset$, then a virtual transition t_{ij} will be added between t_i and t_j , such that $\bullet t_{ij} = P_{out_i}$, $t_{ij} \bullet = P_{in_j}$. If $\bullet P_{in_j} \neq \emptyset$, it means that another activity t_k has added a virtual transition t_{kj} when t_k was added to the model. Then, the only work to do here is to make $P_{out_i} \bullet = \bullet P_{in_j} = t_{kj}$. And activities t_k , t_i and t_j formed an and-join relationship.

Formally, while modeling, if $t_i, t_j \in Relation$ and $Relation.Type = and-join$, then if $\bullet P_{in_j} = \emptyset$, $T = T \cup t_{ij}$, $F = F \cup \{(P_{out_i}, t_{ij}), (t_{ij}, P_{in_j})\}$; else $F = F \cup \{(P_{out_i}, \bullet P_{in_j})\}$. The I/O_WF_NET model for and-join relation is shown in Fig. 3.

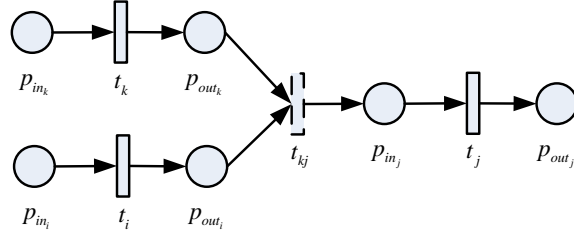


Fig. 3. I/O_WF_NET model for and-join relation

c) The type is *or-join*

If activity t_i , t_k and t_j form an or-join relation, i.e., t_j is the post-activity of both t_i and t_k . Then the virtual transitions t_{ij} and t_{kj} will be added between t_i, t_k and t_j , such that $\bullet t_{ij} = p_{out_i}, \bullet t_{kj} = p_{out_k}, t_{ij} \bullet = t_{kj} \bullet = p_{in_j}$, and the model will be as Fig. 4.

Formally, while modeling, if $t_i, t_j \in Relation$ and $Relation.Type=or-join$, then $T = T \cup t_{ij}, F = F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$.

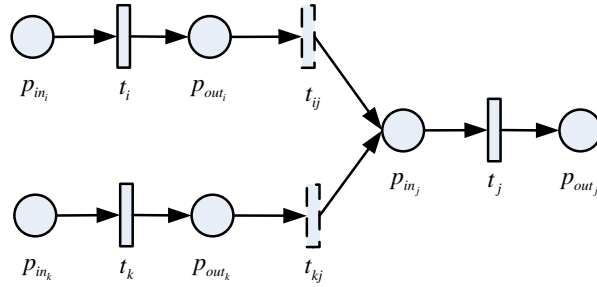


Fig. 4. I/O_WF_NET model for or-join relation

d) The type is *and-split*

If activities t_i and t_j form an and-split relation, the model will depend on whether $p_{out_i}^\bullet = \emptyset$. If $p_{out_i}^\bullet = \emptyset$ then a virtual transition t_{ij} will be added between t_i and t_j , such that $\bullet t_{ij} = p_{out_i}$. If $p_{out_i}^\bullet \neq \emptyset$, it means that a virtual activity t_{ik} has been added while another activity t_k was added to the model. Then, here, we simply let $\bullet p_{in_j} = p_{out_i}^\bullet = t_{ik}$, and activities t_k, t_i and t_j form an and-split a relationship.

Formally, while modeling, if $t_i, t_j \in Relation$ and $Relation.Type=and-split$, then If $p_{out_i}^\bullet = \emptyset$ then $T = T \cup t_{ij}, F = F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$; Else $F = F \cup \{(p_{out_i}^\bullet, p_{in_j})\}$. The I/O_WF_NET model for and-split relation is shown in Fig. 5.

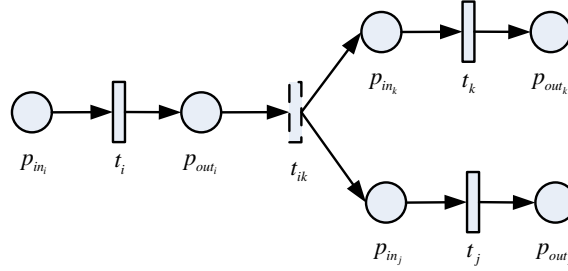


Fig. 5. I/O_WF_NET model for and-split relation

e) The type is *or-split*

If activities t_i , t_j and t_k form an and-split relation, virtual transitions t_{ij} and t_{ik} will directly be added between t_i and t_j , t_k , such that $\bullet t_{ij} = p_{out_i}$, $t_{ij} \bullet = p_{in_j}$, and $\bullet t_{ik} = p_{out_i}$, $t_{ik} \bullet = p_{in_k}$. The model built will be like Fig. 6.

Formally, while modeling, if $t_i, t_j \in Relation$ and $Relation.Type=or-split$, then $T = T \cup t_{ij}$, $F = F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$.

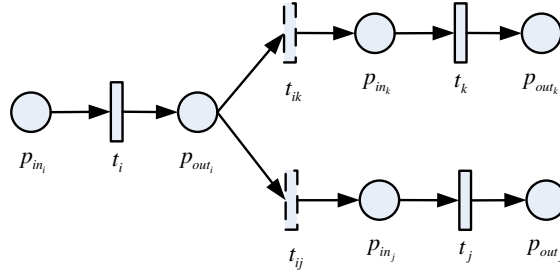


Fig. 6. I/O_WF_NET model for or-split relation

4.2.3. I/O_WF_NET model for start and end activities

a) Start activity

For the activities t_i without pre-activities (e.g., $\bullet p_{in_i} = \emptyset$), a virtual transition t_s will be added before them, and it is called *the start activity*, such that $p_{in_i} \subseteq t_s \bullet$. In addition, for keeping the characteristic of a Petri net, a start place p_s is added before t_s such that $\bullet t_s = p_s$, $p_s \bullet = t_s$, and $\bullet p_s = \emptyset$. The I/O_WF_NET model for three activities without pre-activities is shown in Fig. 7a.

Formally, while modeling, if $t_i \in T$ are activities without pre-activities (e.g., $\bullet p_{in_i} = \emptyset$, where $(p_{in_i}, t_i) \in F$), p_s and t_s are added into the I/O_WF_NET model such that $\bullet p_s = \emptyset$, $p_s \bullet = t_s$, $\bullet t_s = p_s$, $t_s \bullet = p_{in_i}$.

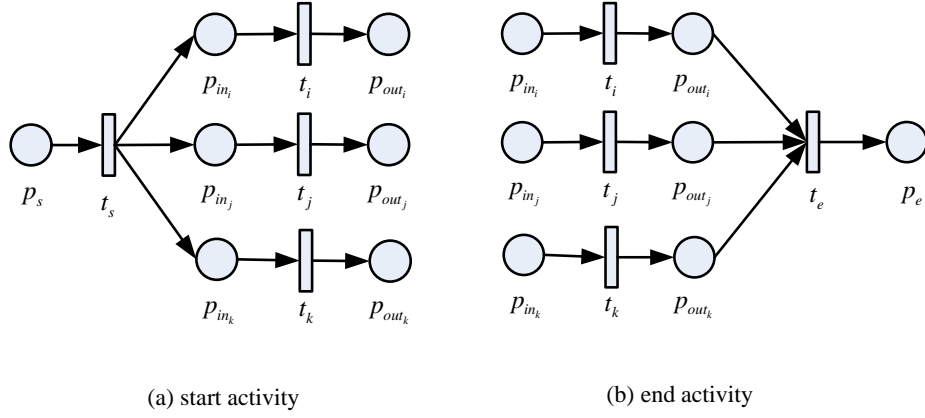


Fig. 7. I/O_WF_NET model for start and end activity

b) End activity

For the activities t_i without post-activities (e.g., $p_{out_i}^{\bullet} = \emptyset$), a virtual transition t_e will be added after them, and it is called *the end activity*, such that $p_{out_i} \subseteq \bullet t_e$. In addition, for keeping the characteristic of Petri net, an end place p_e is added after t_e such that $t_e^{\bullet} = p_e$, $\bullet p_e = t_e$, and $p_e^{\bullet} = \emptyset$, as shown in Fig. 7b.

Formally, while modeling, if $t_i \in T$ are activities without post-activities (e.g., $p_{out_i}^{\bullet} = \emptyset$, where $(t_i, p_{out_i}) \in F$), t_e and p_e are added into the I/O_WF_NET model, such that $\bullet t_e = p_{out_i}$, $t_e^{\bullet} = p_e$, $\bullet p_e = t_e$, $p_e^{\bullet} = \emptyset$.

4.3. Initial marking

The initial marking M_0 of I/O_WF_Net $\Sigma = (P, T, F, M_0)$ satisfies the following:

$$(1) \quad M_0(p) = \begin{cases} 1, & p = p_s, \\ 0 & \text{else.} \end{cases}$$

Based on the modeling details above mentioned, an algorithm for transforming a I/O constrained workflow into the I/O_WF_Net model is given as Algorithm 1.

The time cost of Algorithm 1 mainly depends on the loops in it. For the first one, its time cost is $O(|T|)$. For the second step of modeling the relationship, the time cost will be $O(|Relation|) = O(|T| \times |T|) = O(|T|^2)$. Since the total number of start activities and end activities is less than $|T|$, so the time cost for modeling the start and end activities is $O(|T|)$. The time spend on initial marking of the net is $O(1)$. In conclusion, the time cost of algorithm 1 is $O(|T|^2)$. Taking the workflow in Table 1 as an example, its I/O_WF_Net model after transforming by Algorithm 1 is shown in Fig. 8.

Algorithm 1.

Input. $Workflow = \langle Activity, Input, Output, Relation, f_{AI}, f_{AO} \rangle$

Output. $\Sigma = (P, T, F, M_0)$

begin

$P \leftarrow \emptyset, T \leftarrow \emptyset, F \leftarrow \emptyset, M_0 \leftarrow \emptyset$

$T \leftarrow \text{Activity}$

for $i=1$ to $|T|$ do

$p_{in_i} \leftarrow F_{AI}(t_i) , p_{out_i} \leftarrow F_{AO}(t_i)$

$P \leftarrow P \cup \{p_{in_i}, p_{out_i}\}$

$F \leftarrow F \cup \{(p_{in_i}, t_i), (t_i, p_{out_i})\}$

end for

$\forall t_i, t_j \in T (1 \leq i, j \leq |T|)$ if $t_i, t_j \in \text{Relation}$ then

Switch Relation.Type

Case and-join:

If $\bullet p_{in_j} = \emptyset$ then

$T \leftarrow T \cup t_{ij}$

$F \leftarrow F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$

Else

$F \leftarrow F \cup \{(p_{out_i}, \bullet p_{in_j})\}$

End if

Case and-split:

If $p_{out_i}^\bullet = \emptyset$ then

$T \leftarrow T \cup t_{ij}$

$F \leftarrow F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$

Else

$F \leftarrow F \cup \{(p_{out_i}^\bullet, p_{in_j})\}$

End if

Default:

$T \leftarrow T \cup t_{ij}$

$F \leftarrow F \cup \{(p_{out_i}, t_{ij}), (t_{ij}, p_{in_j})\}$

End Switch

$T_s \leftarrow \{t_i \mid t_i \in T, (p_{in_i}, t_i) \in F \wedge \bullet p_{in_i} = \emptyset\}$

$P \leftarrow P \cup p_s , T \leftarrow T \cup t_s , F \leftarrow F \cup \{(p_s, t_s)\}$

for each $t_i \in T_s$ do $F \leftarrow F \cup \{(t_s, p_{in_i})\}$

$T_e \leftarrow \{t_i \mid t_i \in T, (t_i, p_{out_i}) \in F \wedge p_{out_i} = \emptyset\}$

$P \leftarrow P \cup p_e, T \leftarrow T \cup t_e, F \leftarrow F \cup \{(t_e, p_e)\}$

for each $t_i \in T_e$ do $F \leftarrow F \cup \{(p_{out_i}, t_e)\}$

$M_0(p_s) \leftarrow 1$
 Output $\Sigma = (P, T, F, M_0)$
end

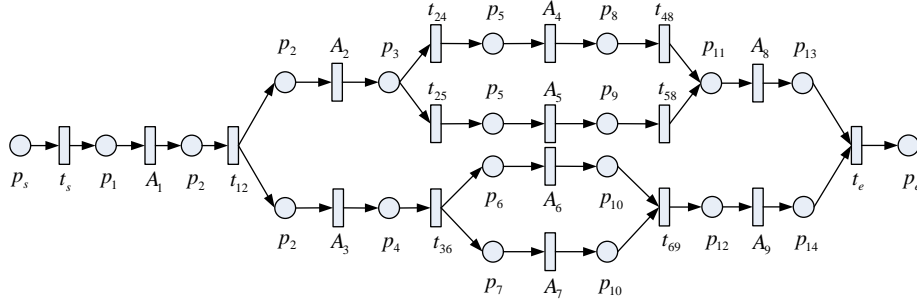


Fig. 8. I/O_WF_NET model for the workflow in Table 1

4.4. Model reduction

The I/O_WF_NET model got by Algorithm 1 may contain many places and transitions, the scale of the model maybe very large, so it will be difficult to analyze and generate an use case. In addition, some different activities may be triggered by the same input, or some different activities may generate the same output, so the model needs to be reduced.

4.4.1. In an *and-join* relation, the preactivities generate the same outputs

If activities t_i, t_k and t_j form an and-join relation, and t_i, t_k have the same outputs, the structure of this case will be as in Fig. 3, where $p_{out_i} = p_{out_k} = p_{out}$. In this case, $t_i, t_k, p_{out_i}, p_{out_k}$ and the relation between them will be deleted first, a new transition t_{ki} and a new place $p_{out_{ki}}$ will be added to the model such that $\bullet t_{ki} = \{p_{in_k}, p_{in_i}\}$, $t_{ki} \bullet = p_{out_{ki}}$, $p_{out_{ki}} \bullet = t_{kj}$, $p_{out_{ki}} = p_{out_k} = p_{out_i}$, the model refined will be shown as Fig. 9. Here, it is the equal of merging t_k, t_i into a new activity t_{ki} , and their outputs are merged as well. The structure of the model is reduced.

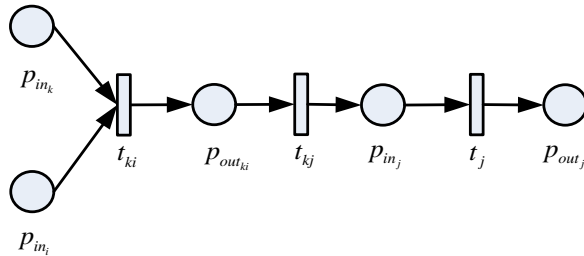


Fig. 9. The refined model for the *and-join* relation

4.4.2. In an *or-join* relation, the preactivities generate the same outputs

If activities t_i, t_k and t_j form an *or-join* relation, and t_i, t_k have the same outputs, the structure of this case will be like the one in Fig. 4, where $p_{out_i} = p_{out_k} = p_{out}$. In this case, $t_{ij}, t_{kj}, p_{out_i}, p_{out_k}$ and the relation between them will be deleted first, a new transition t_{ikj} and a new place $p_{out_{ik}}$ will be added to the model such that $\bullet t_{ikj} = p_{out_{ik}}, t_{ikj}^\bullet = p_{in_j}, t_i^\bullet = t_k^\bullet = p_{out_{ik}}, p_{out_{ik}} = p_{out_i} = p_{out_k}$, the model refined will be shown as Fig. 10.

Here, we just merged the outputs of t_i, t_k , the two activities are still self-existent. While implemented, if any one of their inputs is satisfied, the output $p_{out_{ik}}$ will be got.

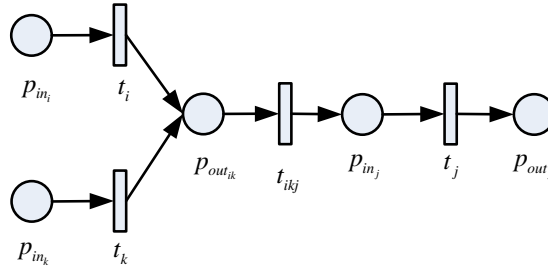


Fig. 10. The refined model for the *or-join* relation

4.4.3. In an *and-split* relation, the inputs of postactivities are the same

If activities t_i and t_k, t_i form an *and-split* relation, and t_k, t_i are triggered by the same inputs, the structure of this case will be like Fig. 5, where $p_{in_k} = p_{in_j} = p_{in}$. In this case, $p_{in_k}, p_{in_j}, t_k, t_j$ and the relation between them will be deleted first, a new transition t_{kj} and a new place $p_{in_{kj}}$ will be added to the model such that $\bullet t_{ik} = p_{in_{kj}}, \bullet t_{kj} = p_{in_{kj}}, t_{kj}^\bullet = \{p_{out_k}, p_{out_j}\}, p_{in_{kj}} = p_{in_k} = p_{in_j}$, the model refined will be shown as Fig. 11. Here, it is equal to merging of t_k, t_i into a new activity t_{kj} , and their inputs are merged as well. The structure of the model is reduced.

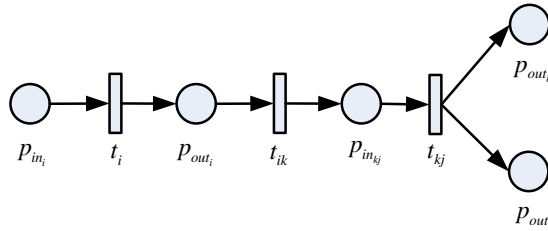


Fig. 11. The refined model for the *and-split* relation

4.4.4. In an *or-split* relation, the inputs of post-activities are the same

If activities t_i and t_k, t_j form an or-split relation, and t_k, t_j are triggered by the same inputs, the structure of this case will be like Fig. 6, where $P_{in_k} = P_{in_j} = P_{in}$. In this case, $t_{ik}, t_{ij}, P_{in_k}, P_{in_j}$ and the relation between them will be deleted first, a new transition t_{ikj} and a new place $P_{in_{kj}}$ will be added to the model, such that $\bullet t_{ikj} = P_{out_i}$, $t_{ikj}^\bullet = P_{in_{kj}}$, $\bullet t_k = \bullet t_j = P_{in_{kj}}$, $P_{in_{kj}} = P_{in_k} = P_{in_j}$, the model refined will be shown as Fig. 12. Here, we just merged the inputs of t_k, t_j , the two activities are still self-existent. While implemented, the workflow will choose one of them to carry on.

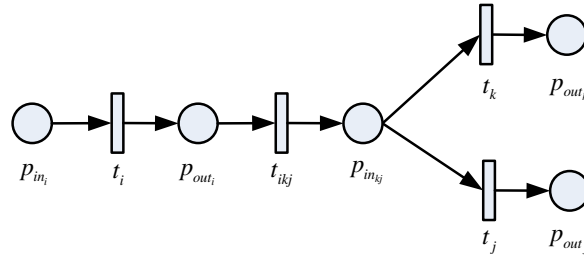


Fig. 12. The refined model for the *or-split* relation

4.4.5. The places besides a virtual transition are the same

If t_{ij} is a virtual transition added to the model by Algorithm 1, and $\bullet t_{ij} = t_{ij}^\bullet$, the structure of this case will be like Fig. 2, where $P_{out_i} = P_{in_j}$. In this case, $t_{ij}, P_{out_i}, P_{in_j}$ and the relation between them will be deleted first, a new place P_{ij} will be added to the model such that $t_i^\bullet = P_{ij}$, $\bullet t_j = P_{ij}$, $P_{ij} = P_{out_i} = P_{in_j}$, the model refined will be shown as Fig. 13. Here, P_{ij} represents both the output of t_i and the input of t_j , the information of the model remain the same.

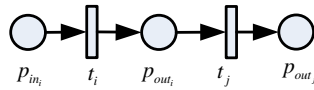


Fig. 13. The refined model if the places besides a virtual transition are the same

The model after being refined from (1) to (5) may not be the simplest, but the scale of the model has become much smaller.

The reduction result of the I/O_WF_Net model in Fig. 13 is shown in Fig. 14, and Table 2 provides comparison of the I/O_WF_Net model before and after reduction, from which it is easy to get the conclusion that the scale and complexity have reduced a lot.

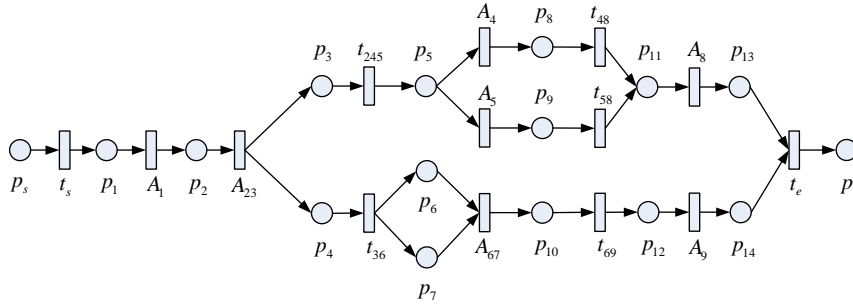


Fig. 14. The refined result of I/O_WF_Net model in Fig. 13

Table 2. Comparison of i/o_wf_net model before and after reduction

Item	Before reduction	After reduction
Number of activities	9	7
Number of places	20	16
Number of transitions	18	14

5. Conclusion and future work

Workflow systems have been used in still more areas and the validity and reliability of them are becoming more and more important. The research on workflow systems testing is also developing more and more. After the basic concept of Petri net is introduced, we present a testing oriented workflow modeling approach.

The main contributions of this paper include the following:

1) A formal definition for workflow net constrained by inputs and outputs is performed, and every component of the definition is analyzed and described in details.

2) A I/O_WF_Net model and a modeling approach for workflow constrained by inputs and outputs are presented.

3) An algorithm for transforming a workflow net constrained by inputs and outputs to I/O_WF_Net is given, and the complexity of the algorithm is analyzed.

4) Based on analyzing of I/O_WF_Net, five specific situations for reducing the model are discussed, and we show how to apply this approach through a case study.

At present, we only did a research on the workflow modeling approach for testing. The work of this paper opens the door to future research on the following three subjects: firstly, the paper only discussed the activities which have only one input and one output. In future, the research on a modeling and reducing method for activities that have multiple inputs and multiple outputs must be carried on. Secondly, based on the modeling approach presented in this paper, the test case generation method should also be given. Thirdly, research on test coverage oracle for completeness must be taken

Acknowledgment: This work is supported by the National High Technology Research and Development Program of China (No 2009AA01Z402), China Postdoctoral Science Foundation (No 20110491843) and the Natural Science Foundation of Jiangsu Province, China (Grant Nos BK2012059, BK2012060). Resources of the PLA Software Test and Evaluation Centre for Military Training are used in this research.

References

1. Wang, H., Q. Zeng. Modelling and Analysis for Workflow Constrained by Resources and Nondetermined Time-An Approach Based on Petri Nets. – IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, Vol. **38**, July 2008, No 4, pp. 802-816.
2. Pang, S., C. Jiang. Workflow Performance Analysis Based on Invariant Decomposition Algorithm. – Chinese Journal of Computers, Vol. **33**, May 2010, No 5, pp. 908-918.
3. Yu, Y., Y. Tang et al. Temporal Workflow Process Model and Its Soundness Verification. – Journal of Software, Vol. **21**, June 2010, No 6, pp. 1233-1253.
4. Merdan, M., T. Moser et al. Simulation of Workflow Scheduling Strategies Using the MAST Test Management System. – In: Proc of 10th Intl. Conf. on Control, Automation, Robotics and Vision, 2008, pp. 1172-1177.
5. Stratan, C., A. Iosup et al. A Performance Study of Grid Workflow Engines. – In: Proc. of 9th Grid Computing Conference, 2008, pp. 25-32.
6. Wang, J., D. Rosca, W. Tepfenhart et al. Dynamic Workflow Modelling and Analysis in Incident Command Systems. – IEEE Trans. Syst., Man., Cybern. A, Syst., Humans, Vol. **38**, September 2008, No 5, pp. 1041-1055.
7. Du, Y., C. Jiang, M. Zhou. Modelling and Analysis of Real-Time Cooperative Systems Using Petri Nets. – IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, Vol. **37**, September 2007, No 5, pp. 643-654.
8. Kim, K., H. Ahn, C. Kim. Performance Estimations of Clustered Workflow Architectures. – In: Proc. of 4th Annual ACIS International Conference on Computer and Information Science (ICIS'05), 2005.
9. Zheng, J., Q. Chen, C. Qi. Verification and Reduction of Cyclic Structure In Workflow Model. – In: Proc. of Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, August 2005, pp. 1487-1492.
10. Li, J., Y. Fan, M. C. Zhou. Performance Modeling and Analysis of Workflow. – IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, Vol. **34**, March 2004, No 2, pp. 229-242.
11. Li, J., Y. Fan, M. C. Zhou. Timing Constraint Workflow Nets for Workflow Analysis. – IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, Vol. **33**, March 2003, No 2, pp. 179-192.
12. Karniel, A., Y. Reich. Formalizing a Workflow-Net Implementation of Design-Structure-Matrix-Based Process Planning for New Product Development. – IEEE Trans. Syst., Man., Cybern. A, Syst., Humans, Vol. **47**, May 2011, No 3, pp. 476-491.
13. Nichols, J., H. Demirkan, M. Goul. Autonomic Workflow Execution in the Grid. – IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., Vol. **36**, May 2006, No 3, pp. 353-364.
14. Van Der Aalst, W. M. P., T. Weijters, L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. – IEEE Trans. Knowl. Data Eng., Vol. **16**, September 2004, No 9, pp. 1128-1142.
15. Janssens, G. K., B. W. Jan Verelst, B. Weyn. Techniques for Modeling Workflows and Their Support for Reuse. – In: Business Process Management: Models, Techniques, and Empirical Studies. W. Van Der Aalst, Ed. Berlin, Germany, Springer-Verlag, 2000, pp. 1-15.
16. Hwang, G., C. Lin et al. A Framework and Language Support for Automatic Dynamic Testing of Workflow Management Systems. – In: Proc. of 2009 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2009), pp. 139-146.
17. Karam, M., W. Keiروز, R. Hage. An Abstract Model for Testing MVC and Workflow Based Web Applications. – In: Proc. of Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services. (AICT/ICIW 2006).
18. Quan, L., X. Lin, J. Wang. An Automatic and Scalable Testing Tool for Workflow Systems. – In: Proc. of 3rd International Conference on Grid and Pervasive Computing – Workshops(GPC 2008), pp. 75-80.
19. Bartz, R. Workflow for Automotive Test Data Analysis Based on Petri Nets and Stored by ASAM ODS.
20. Petri, C. A. Kommunikation mit automaten. Ph.D. Dissertation, Inst. Instrumentelle Math., Bonn, Germany, 1962.
21. Reisig, W. Petri Nets: An introduction. – In Monographs in Theoretical Computer Science, An EATCS Series, Vol. **4**, Springer-Verlag, Berlin, Germany, 1985.
22. Murata, T. Petri Nets: Properties, Analysis and Applications. – Proc. of IEEE, Vol. **77**, April 1989, No 4, pp. 541-580.