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    yaoyi226@aliyun.com    hs0317@163.com
zhengpingren@gmail.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 outputs. Section 4 introduces the R/NT_WF_Net model and the modeling approach for a workflow constrained by inputs and outputs. 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 \(\langle \text{Activity}, \text{Input}, \text{Output}, \text{Relation}, f_{AI}, f_{AO} \rangle\), where

1) \(\text{Activity} = \{\text{activity}_1, \text{activity}_2, \ldots, \text{activity}_k\}\) is the activity set of a workflow.

2) \(\text{Input} = \{\text{input}_1, \text{input}_2, \ldots, \text{input}_m\}\) is the input set of a workflow.

3) \(\text{Output} = \{\text{output}_1, \text{output}_2, \ldots, \text{output}_n\}\) is the output set of a workflow.

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

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

6) \(f_{AO} : \text{Activity} \rightarrow \rho(\text{Output})\) is the output function of a workflow where \(\rho(\text{Output})\) 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 \(\text{Activity}\) includes all the activities in the workflow.

2) The set \(\text{Input}\) includes all the input elements in the workflow.

3) The set \(\text{Output}\) includes all the output elements in the workflow.

4) The set \(\text{Relation}\) defines the relations between activities and their types. \(\forall \text{activity}_1, \text{activity}_2 \in \text{Activity}, \text{if} (\text{activity}_1, \text{activity}_2) \in \text{Relation}, \text{then} \text{activity}_2\) cannot started until \(\text{activity}_1\) is finished, and \(\text{activity}_2\) is called the pre-activity of \(\text{activity}_2\), \(\text{activity}_1\) is called the post-activity of \(\text{activity}_1\). 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 $i \in \text{Activity}$, input $i \subseteq \text{Input}$, if $f_{i}(\text{activity}_i) = \text{input}_i$, then it means that the execution of activity$_i$ needs an input$_i$; if $f_{i}(\text{activity}_i) = \emptyset$ then the execution of activity$_i$ does not need any input.

6) For activity $i \in \text{Activity}$, output $i \subseteq \text{Output}$, if $f_{oa}(\text{activity}_i) = \text{Output}_i$ then the implementation of activity$_i$ will output output$_i$; if $f_{oa}(\text{activity}_i) = \emptyset$ then the implementation of the activity$_i$ will not output anything.

Table 1 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>
<thead>
<tr>
<th>Activity</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Preactivities</th>
<th>Relation type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>$p_1$</td>
<td>$p_2$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$A_2$</td>
<td>$p_2$</td>
<td>$p_3$</td>
<td>$A_1$</td>
<td>and-split</td>
</tr>
<tr>
<td>$A_3$</td>
<td>$p_2$</td>
<td>$p_4$</td>
<td>$A_1$</td>
<td>and-split</td>
</tr>
<tr>
<td>$A_4$</td>
<td>$p_5$</td>
<td>$p_8$</td>
<td>$A_2$</td>
<td>or-split</td>
</tr>
<tr>
<td>$A_5$</td>
<td>$p_5$</td>
<td>$p_6$</td>
<td>$A_2$</td>
<td>or-split</td>
</tr>
<tr>
<td>$A_6$</td>
<td>$p_6$</td>
<td>$p_{10}$</td>
<td>$A_3$</td>
<td>and-split</td>
</tr>
<tr>
<td>$A_7$</td>
<td>$p_7$</td>
<td>$p_{10}$</td>
<td>$A_3$</td>
<td>and-split</td>
</tr>
<tr>
<td>$A_8$</td>
<td>$p_{11}$</td>
<td>$p_{13}$</td>
<td>$A_4, A_5$</td>
<td>or-join</td>
</tr>
<tr>
<td>$A_9$</td>
<td>$p_{12}$</td>
<td>$p_{14}$</td>
<td>$A_6, A_7$</td>
<td>and-join</td>
</tr>
</tbody>
</table>

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 $\forall p \in \text{rst} , M(P) > 0$, and that is written as $M[t >$;
2) $\forall M_1, M_2, M_i \leq M_2$ 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., $i^* = \emptyset$; $o$ is an end place, i.e., $o^* = \emptyset$.

If a new transition $t$ is added to $PN$, such that $t^* = \{o\}$, $i^* = \{i\}$, 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 $p_{in} \in i^* t$.
5) $\forall t \in T$, $p_{out} \in P_{out}$, the execution of $t$ will output $p_{out}$ iff $p_{out} \subseteq t^*$.

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.

In Fig. 1, $t_i$ represents activity $i$, $P_{in}, P_{out}$ are the input and output of $t_i$. If $P_{in}$ contains tokens, it means that the input of $t_i$ is satisfied and the activity can be
implemented; if $P_{out}$ contains tokens, then it means that the implementation of $t_i$ is finished, and it produced $P_{out}$. Formally, activity $t_i$ can be denoted by $[P_{in}, t_i, P_{out}]$.

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 $P_{out}$ 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 $P_{in}$ 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 $(t_i, t_j) \in \text{Relation}$, e.g., activity $[P_{in}, t_j, P_{out}]$ is the post-activity of activity $[P_{in}, t_i, P_{out}]$, 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 $t_j$ is added between them, such that $i_{ij} = P_{out}$ and $j_{ij} = P_{in}$. As in Fig 2, $t_j$ 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](image)

b) The type is and-join
If the relation type between activity $t_i$ and $t_j$ is and-join, then the model builds here depends on whether $i_{in} = \emptyset$ or not. If $i_{in} = \emptyset$, then a virtual transition $t_j$ will be added between $t_i$ and $t_j$, such that $i_{ij} = P_{out}$ and $j_{ij} = P_{in}$. If $i_{in} \neq \emptyset$, it means that another activity $t_k$ has added a virtual transition $t_j$ when $t_k$ was added to the model. Then, the only work to do here is to make $i_{out} = i_{out} \cup \{t_j\}$. And activities $t_k, t_i$ and $t_j$ formed an and-join relationship.

Formally, while modeling, if $t_i, t_j \in \text{Relation}$ and $\text{Relation.Type}=\text{and-join}$, then if $i_{in} = \emptyset$, $T = T \cup t_j$, $F = F \cup \{(P_{out}, t_j), (t_j, P_{in})\}$; else $F = F \cup \{(P_{out}, i_{in})\}$. The I/O_WF_NET model for and-join relation is shown in Fig. 3.
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 \( t_y \) and \( t_{ij} \) will be added between \( t_i, t_k \) and \( t_j \), such that \( t_y = p_{out}, t_{ij} = p_{out}, t_y^* = t_{ij}^* = p_{in} \), and the model will be as Fig. 4.

Formally, while modeling, if \( t_i, t_j \in Relation \) and \( Relation.Type = \text{or-join} \), then
\[
T = T \cup t_y, \quad F = F \cup \{(p_{out}, t_y), (t_y, p_{in})\}.
\]

![Fig. 3. I/O_WF_NET model for and-join relation](image)

![Fig. 4. I/O_WF_NET model for or-join relation](image)

d) The type is *and-split*

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

Formally, while modeling, if \( t_i, t_j \in Relation \) and \( Relation.Type = \text{and-split} \), then
\[
T = T \cup t_y, \quad F = F \cup \{(p_{out}, t_y), (t_y, p_{in})\}; \quad \text{Else} \quad F = F \cup \{(p_{in}^*, p_{in})\}.
\]

The I/O_WF_NET model for and-split relation is shown in Fig. 5.
e) The type is or-split

If activities \( t_i \), \( t_j \) and \( t_k \) form an and-split relation, virtual transitions \( t_j \) and \( t_k \) will directly be added between \( t_i \) and \( t_j \), \( t_k \), such that \( \text{out}_{ijt} = p_{out_i} \), \( \text{out}_{ikt} = p_{out_k} \), and \( \text{out}_{ikt} = p_{out_k} \). The model built will be like Fig. 6.

Formally, while modeling, if \( t_i, t_j \in \text{Relation} \) and \( \text{Relation.Type} = \text{or-split} \), then

\[
T = T \cup t_j, \quad F = F \cup \{(p_{out_i}, t_j), (t_j, p_{in_i})\}.
\]

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., \( \text{in}_{ip} = \emptyset \)), a virtual transition \( t_s \) will be added before them, and it is called the start activity, such that \( \text{in}_{isp} \subseteq t_s^* \).

In addition, for keeping the characteristic of a Petri net, a start place \( p_s \) is added before \( t_i \) such that \( \text{in}_{isp} = p_s \), \( \text{in}_{tsp} = t_s \), and \( \text{in}_{tsp} = \emptyset \). 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., \( \text{in}_{lip} = \emptyset \), where \( \text{in}_{lip}, t_i \in F \)), \( p_s \) and \( t_s \) are added into the I/O_WF_NET model such that \( \text{in}_{lip} = \emptyset \), \( \text{in}_{tsp} = t_s \), \( \text{in}_{tsp} = p_s \), and \( \text{in}_{tsp} = p_{in_i} \).
b) End activity

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

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

4.3. Initial marking

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

\[
M_0(p) = \begin{cases} 
1 & \text{if } p = p_s, \\
0 & \text{else.}
\end{cases}
\]

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=$\langle Activity, Input, Output, Relation, f_d, f_o \rangle$
Output. $\sum = (P, T, F, M_0)$

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

$T \leftarrow \text{Activity}$

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

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

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

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

end for

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

Swich Relation.Type

Case and-join:

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

$T \leftarrow T \cup t_j$

$F \leftarrow F \cup \{(p_{out_j}, t_j), (t_j, p_{in_j})\}$

Else

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

End if

Case and-split:

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

$T \leftarrow T \cup t_j$

$F \leftarrow F \cup \{(p_{out_j}, t_j), (t_j, p_{in_j})\}$

Else

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

End if

Default:

$T \leftarrow T \cup t_j$

$F \leftarrow F \cup \{(p_{out_j}, t_j), (t_j, p_{in_j})\}$

End Swich

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

$P \leftarrow P \cup p_i , T \leftarrow T \cup t_i , F \leftarrow F \cup \{(p_i, t_i)\}$

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

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

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

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

36
\[ M_0(p_i) \leftarrow 1 \]
\[
\text{Output } \Sigma = (P, T, F, M_o) 
\]
\[ \text{end} \]

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, \( t_i, t_k, p_{out}, p_{out_i} \) and the relation between them will be deleted first, a new transition \( t_{ki} \) and a new place \( p_{out_k} \) will be added to the model such that \( t_{ki} = \{p_{in_i}, p_{in_k}\}, p_{out_k} = p_{out} \), the model refined will be shown as Fig. 9. Here, it is the equal of merging \( t_i, t_k \) 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, $t_j, t_{ij}, p_{out_i}, p_{out_k}$ and the relation between them will be deleted first, a new transition $t_{ij}$ and a new place $p_{out_{ij}}$ will be added to the model such that $t_{ij} = p_{out_{ij}}$, $t_j = t_{ij} = p_{out_{ij}}$, $p_{out_{ij}} = p_{out_i} = p_{out_k}$, 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 $p_{out_{ij}}$ will be got.

![Fig. 10. The refined model for the or-join relation](image)

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

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

![Fig. 11. The refined model for the and-split relation](image)
4.4.4. In an or-split relation, the inputs of post-activities are the same
If activities \( t_i \) and \( t_k, t_l \) form an or-split relation, and \( t_k, t_l \) are triggered by the same inputs, the structure of this case will be like Fig. 6, where \( p_{in_i} = p_{in_j} = p_{in} \). In this case, \( t_k, t_l, p_{in_k}, p_{in_l} \) and the relation between them will be deleted first, a new transition \( t_{ij} \) and a new place \( p_{out_k} \) will be added to the model, such that \( \cdot t_{ij} = p_{out_k}, t_{ij} = p_{in_k}, p_{out_j} = p_{in_j} \), the model refined will be shown as Fig. 12. Here, we just merged the inputs of \( t_k, t_l \), 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](image)

4.4.5. The places besides a virtual transition are the same
If \( t_y \) is a virtual transition added to the model by Algorithm 1, and \( \cdot t_y = \cdot t_y \), the structure of this case will be like Fig. 2, where \( p_{in_y} = p_{in} \). In this case, \( t_y, p_{in_y}, p_{in} \) and the relation between them will be deleted first, a new place \( p_y \) will be added to the model such that \( \cdot t_y = p_y, t_y = p_{in_y}, p_y = p_{out_y} = p_{in_y} \), the model refined will be shown as Fig. 13. Here, \( p_y \) 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](image)

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.
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

19. Bartz, R. Workflow for Automotive Test Data Analysis Based on Petri Nets and Stored by ASAM ODS.