Layered Workflow Process Model Based on Extended Synchronizer

Ni Gang
School of Management, Guangdong University of Technology, Guangzhou 510520, China
School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China

Zhang Bixi and Wu Pangpang
School of Management, Guangdong University of Technology, Guangzhou 510520, China
Email: nishaogang@gdut.edu.cn

Abstract—The layered workflow process model provide a modeling approach and analysis for the key process with Petri Net. It not only describes the relation between the process of business flow and transition nodes clearly, but also limits the rapid increase in the scale of libraries, transition and directed arcs. This paper studies the process like reservation and complaint handling information management system, especially for the multi-mergence and discriminator patterns which can not be directly modeled with existing synchronizers. Petri Net is adopted to provide formalization description for the workflow patterns and the relation between Arcs and weight class are also analyzed. We use the number of in and out arcs to generalize the workflow into three synchronous modes: fully synchronous mode, competition synchronous mode and asynchronous mode. The types and parameters for synchronization are added to extend the modeling ability of the synchronizers and the synchronous distance is also expanded. The extended synchronizers have the ability to terminate branches automatically or activate the next link many times, besides the ability of original synchronizers. By the analyses on cases of the key business, it is verified that the original synchronizers can not model directly, while the extended synchronizers based on Petri Net can provide modeling for multi-mergence and discriminator modes.

Index Terms—Workflow Process; Petri Nets; Synchronizer; Arcs; Synchronous Distance

I. INTRODUCTION

When designing a specific application system, we need modeling for the workflow model and workflow process model is the core of workflow model. Therefore, if workflow process model is clearly described, it will guarantee the smoothness of each workflow samples without deadlock and make the workflow model set up a feasible, complete and effective workflow management system [1]. At present, there are many modeling tools of workflow process model. Among them, Petri Net is a system model which is suitable for describing concurrent characteristics. It has become a rational modeling tool in workflow [2, 3, 4]. However, in practical implementation, basic Petri Net usually has more complicated problems in modeling [5], the workflow net modeling also has indirect and unclear understanding in its process, causing some error process appear. In order to solve the problems during modeling, references [6, 7] divide the workflow process model into logic layer and semantic layer. Logic layer of workflow process model expresses logic structure in the process and it clearly abstracts the main parts in workflow process. Semantic layer of workflow process model aims at specific cases in workflow. Based on dominant content, it selects practical path of cases in path framework to be used to solve conflicts in workflow logic. Logic layer of hierarchical workflow process model contains activities, transference and connector. Connector can be expressed as workflow pattern in abstraction and workflow pattern can be expressed as synchronizer. That is, the synchronizer can become modeling workflow pattern. References [8, 9] adopt the principle between logic and semantic division and it combines 21 workflow patterns into 9 workflow patterns. References [10] study above workflow patterns with description method of synchronizer. Synchronizer description method is a branch of universal Petri theories. On one hand, practical business process usually displays synchronization phenomenon in different stages. Therefore, synchronizer can connect all activities into an entirety to show a complete process structure with advantages of simplicity and high efficiency. On the other hand, synchronizer also undertakes the advantages of Petri net in strong expression and abstraction when describing the workflow.

Based on the research on modeling theory by inspecting and comparing the works of domestic and overseas, this paper is aiming at several core problems of realizing the target of workflow management system which involves workflow meta-model, workflow process model, etc. On the basis of existing achievements, we study the expansibility of workflow meta-model, the problems in modeling workflow process models and realization of workflow management system. During the studying process of the workflow model, this paper first analyzes the model complexity of modeling with two methods: Petri net and layer modeling based on synchronizer. It confirms that workflow layer modeling theory based on Petri net synchronization cannot only solve the problems of element explosion growth on...
database, changes, etc. during Petri net modeling. Meanwhile, its logic layer can effectively and clearly abstract main parts of workflow process model. The dominant contents involving specific cases are described in semantic layer to promote the complete workflow process model to be direct and clear. Secondly, contrapositing to the problems on research of logic layer on hierarchical workflow process model, we propose the expansible synchronizer theory for Petri net. Current Petri net synchronizer has problems that some workflow patterns cannot directly be modeled on logic layer modeling of workflow process models. Through extended original synchronizers’ parameters and types, it can directly be modeled for all workflow patterns. Compared to synchronizer before extension, it improves the ability of modeling discriminator and multi-synchronization. Meanwhile, it also offers specific implementation algorithm of this expansible synchronizer. It is proved to have modeling ability of extended synchronizer with specific cases. It also improves the modeling ability of synchronizer on logic layer of workflow process model. On the basis of above theoretical research, we analyze the demand in hotel business operation support system. It designs the workflow meta-model of one large-scaled hotel information management system. Formalization description of the core layers in this systematic workflow meta-model and process model, organization model and information model of initial extension layer are also provided. The key functions of management information system-workflow engine with workflow process definition and implementation algorithm of workflow pattern is realized. It successfully introduces the workflow management system to solve some problems in hotel business management.

II. LAYERED WORKFLOW PATTERNS BASED ON PETRI NET

A. Complexity Analysis of Layered Workflow Process

We list an example of single process model which is modeled by layered workflow process model. It includes three processed. There are three departments handling or transmitting the fault or complaint reports. The constructed single process model is shown as figure 1.

Figure 1. An example of 3-node model based on layered workflow process model

Its formalization is defined as

\[ \sum = (P, V, T; F, K, W, R, W_r, M_F, M_o) \]

\[ P = \{p_0, p_1, p_2, p_3\} \]

is the library set, representing the synchronizer of logic layer;

\[ V = \{x, y\} \]

is variables set, representing the rules of semantic layer;

\[ T = \{t_0, t_1, t_2, t_3, t_4, t_5\} \]

is transition set, representing the activities of logic layer;

\[ F = \{(p_0, t_0), (l_0, p_1), (p_1, t_1), (p_2, t_2), (p_3, t_3), (p_4, t_4), (p_5, t_5)\} \]

is flow relation set, representing the arc expressions.

\[ K = 1 \]

denotes there is one token circulating every time in the network;

\[ W = 1 \]

denotes the weight of each arc is 1;

\[ R = \{(x, t_1), (x, t_3), (y, t_4), (y, t_5)\} \]

is the reading relation set;

\[ W_r = \{(y, t_1), (y, t_3)\} \]

is the writing relation set;

\[ M_F = \text{status + guard + body} \]

is the transition mark of workflow. status is status function, guard is whistle function and body is body function;

\[ M_o = (M_{w_p}, M_{w_o}) \]

is the original mark of \[ \sum \].

In this case, the basic workflow logic is very simple. But if adopting the style of original Petri Net, it needs to add monitoring library and related transition to judge the circulation path. This will cause extremely complicated structure and lead to bad simulation and control of the business process. When the semantics are introduced, the model structure has greatly simplified the basis of original Petri Net and it does not weaken the description ability of special business. The business process becomes clear and more close to actual cognitive and management practices.

So the layered workflow process model can clearly describe the 3-node cycling processes. All the libraries, transitions and scales of directed arcs do not make explosive growth. During the logic layer modeling of the layered workflow process, though the synchronizer can provide models for most workflow patterns, we cannot merge the two workflow patterns: modeling multiple mergers and discriminators. Therefore, we need further research on the extended synchronizer to solve this problem in the following sectors.

B. Formalization Description of the Workflow

References [8] and [9] reduce the traditional workflow modes to 9 modes which are commonly used on logic layer. They are Sequence, AND-SPLIT, AND-JOIN, XOR-SPLIT, OR-SPLIT, XOR-JOIN, OR-JOIN, MULTI-MERGENCE, and N OUT OF N JOIN. Petri nets is used to describe these 9 modes to analyze into-arc number and out-arc number of each mode, so as to prepare for workflow mode classification based on the synchronous distance. Here the element \[ P \] is used to express the connector and element \[ T \] is used to express the activity [11]. These 9 modes are analyzed as followings:

Sequence, AND-SPLIT, XOR-SPLIT and OR-Split all start one or many paralleled branches after one thread started. The difference is whether the activation branch
number needs activation conditions. Figure 2 shows the description of using Petri nets. $N$ denotes the number of branches which are ready to be activated. $a$ denotes the weight of arc which shows the number of branches in follow-up links that needed to be activated.

**SEQUENCE:** $n = a = 1$

**XOR-SPLIT:** $n > a = 1$

**Selection-SPLIT:** $n > a > 1$

**AND-SPLIT:** $n = a > 1$

![Figure 2. Formalization of sequence AND split and OR split](image)

XOR-SPLIT, OR-SPLIT, AND-JOIN belong to the activities of next link which is activated after all the activated branches aggregate at the aggregation point. The difference lies in the number of branches at the aggregation point. MULTI-MERGENCE belongs to the activity of next link which will be activated when each activated branch arrives at the aggregation point. It does not need the arrival of all the activated branches. We use Petri nets for description as figure 3. $m$ denotes the number of activated in-arcs of place $P$; $n$ denotes the number of all the in-arcs; $a$ denotes the weight of out-arc.

**XOR-SPLIT:** $n > m = a = 1$

**OR-SPLIT:** $n > m = a > 1$

**AND-JOIN:** $n = m = a > 1$

**MULTI-MERGENCE:** $n \geq m > a = 1$

Since $m > a$, the follow-up links will be activated many times.

![Figure 3. Formalization of XOR join, OR join and multiple mergence](image)

Formalization of discriminator is shown as figure 4. $m$ denotes the number of activated in-arcs of place $p_2$; $n$ denotes the number of all the in-arcs of $p_1$; The false place composed by $p_1$ and $p_2$ denotes the aggregation point. $p_1$ controls the times of triggering in the next link and $p_2 = 1$. When $a$ branches arrive $p_2$ the next link will be triggered. Once it is triggered, the follow-up branches can not be triggered if $p_1 = 0$.

![Figure 4. Formalization of the discriminator](image)

As is known from above descriptions, the common ground of these modes is that: several in-arcs will activate the branch of next link after they are aggregated according to the aggregation rules. Any mode will adopt the descriptions like in, out arc, and weight, which is shown in table 1. The following symbols are defined first for easy analysis:

- Out-arc set of $P$ : $Out(P) = \{a' = (P, t) \mid t \in T\}$
- Weight of in-arc: $W(a)$
- Weight of out-arc: $W(a')$
- Number of in-arcs: $num(\text{In}(P))$
- Number of out-arcs: $num(\text{Out}(P))$
- Number of activated in-arcs: $\text{Act}_num(\text{In}(P))$
- Number of activated out-arcs: $\text{Act}_num(\text{Out}(P))$
- Capacity of $P$: $C(P) = a_1 \times a_2$

$C(P) = a_1 \times a_2$ describes the weighted synchronous distance of synchronous relation between two links [12,13] and it is a directed synchronization. According to above analysis, we can generalize 9 types of workflow modes into three synchronous modes: complete synchronization, competitive synchronization and non synchronization.

III. SYNCHRONIZER EXTENSION BASED ON PETRI NETS

A. Synchronizer Evaluation and Extension

Reference [14] segregates the logic, rules and management and provides modeling with different models respectively. In the logic layer, the logic relation among the neighbour links of synchronizers, that is, the connectors are basic structure for three-layered model sharing [15]. Then we analyze its features as the follows.

As is shown in figure 5, assume $T_1$ select $a_1$ tasks and $T_2$ select $a_2$ tasks. Then the weighted synchronous distance $\delta(T_1, T_2, a_1, a_2)$ between $T_1$ and $T_2$ is the capacity $C(P) = a_1 \times a_2$. The synchronizer $P = (T_1, T_2, a_1, a_2)$ is not known until it is executed. So we can use variables to denote $P = (T_1, T_2, (x, y), 1 \leq x \leq m_1, 1 \leq y \leq m_2)$. The feature of the synchronizer is:

- $C(P) = \text{Act} _num(\text{In}(P)) \times \text{Act} _num(\text{Out}(P))$
- $W(a') = \text{Act} _num(\text{In}(P))$
The capacity of $P$ has described the synchronous distance of neighbour links. According to our analysis, the synchronizer belongs to complete synchronous mode. It can clearly be seen that this synchronizer can not fully represent non synchronous mode, that is, competitive synchronization and non synchronization.

The weighted synchronous distance equals to the product of weights of out-arc and in-arc. $a_2$ equals to the count of branches activated in subsequent links and $a_1$ equals to the count of activated in-arcs. In complete synchronous mode, the weight of in-arcs is the same with activated out-arcs and the weight of out-arcs is the same with activated in-arcs. The synchronizer will keep waiting until all the activated $a_i$ arrive. Then it merges the documents and passes them to subsequent activated links each [16]. But it is not the case in competitive synchronization and non synchronization: The synchronizer will begin merging and copying when part of the in-arcs arrives.

Then we need to add parameters for synchronization control on the basis of the definition to original synchronizer. So the synchronizer can control different synchronous behaviors according to synchronized control parameters, which can provide modeling for different synchronous modes. The extended method is given below:

$$ P = (T_1, T_2, (a_1, a_2), (m_1, m_2), \{\text{type}, \text{count}\}) $$

$$ T_i = \{t_{11}, t_{12}, \ldots, t_{i1}\}, \quad T_2 = \{t_{21}, t_{22}, \ldots, t_{2m2}\} $$

$a_i$ denotes the executed task selected in link $T_i$; $a_2$ denotes the executed task selected in link $T_2$. $m_1$ denotes the number of in-arcs of the synchronizer; $m_2$ the number of in-arcs of the synchronizer.

Type denotes the synchronization mode and type $\in$ labelSet

Full_SYN, Compet_SYN, Non_SYN respectively denotes complete synchronization, competitive synchronization and non synchronization.

$$ \text{labelSet} = \{\text{Full_SYN, Compet_SYN, Non_SYN}\} $$

count denotes the number of branches needed synchronizing and

$$ \text{count} = \{x \in \{\ast\} \cup \{1, 2, 3, \ldots\} \} $$

* denotes all the activated branches, that is, count = $a_i$.

The extended synchronous distance is:

$$ \delta(T_i, T_j) = \begin{cases} 
\text{count} \times a_2, & \text{if count} \neq \ast \\
\text{count} \times a_1, & \text{else}
\end{cases} $$

B. Modeling by Synchronous Mode

(a) Complete synchronization mode

$$ P = (T_1, T_2, (a_1, a_2), (m_1, m_2), (\text{Full_SYN}, \ast)) $$

$$ \delta(T_i, T_j) = a_1 \times a_2 $$

(b) Competitive synchronization mode

$$ P = (T_1, T_2, (a_1, a_2), (m_1, m_2), (\text{Comp_SYN}, \text{count})) $$

$$ \delta(T_i, T_j) = \text{count} \times a_2, \text{count} \geq 1 $$

(c) Non synchronization mode

$$ P = (T_1, T_2, (a_1, a_2), (m_1, m_2), (\text{Non_SYN}, \text{1})) $$

$$ \delta(T_i, T_j) = 1 \times a_2 $$

The extended synchronizer has extended the synchronous types and synchronous control parameters to model all the synchronous modes. Compared to the original synchronizers, we provide improvement in the following aspects:

(1) When definite the extended synchronizer, we provide the count of in-arcs and out-arcs explicitly. It is easy to design the follow-up synchronous algorithm.

(2) We add the count of branches needed to be synchronized, which control the time of synchronizers to work. If count = $\ast$, it explains that the synchronization will not begin until all the branches arrive; if count = 1, then the next link will be activated when one branch first arrived.

(3) The synchronous type parameter type is adopted by us to control the synchronous behaviors of synchronizers together with count. The operation merging can activate the follow-up links many times, but there is only one for the discriminator. So the extended synchronizer can control the times of activation for follow-up links, by extending the synchronous types. It corrects the computation of synchronous distance of the synchronizer. The synchronous distance still adopts the weighted synchronous distance but parameter count is added and the weight of out-arc is decided by count. Thus, the synchronous distance will be well corrected as the definition of extended synchronizer. The workflow engine calls different synchronous algorithms according to the synchronous types, to acquire different synchronous modes.
C. Algorithm Realization

Variable definition:

\( p \) : synchronizer

\( t \) : sub-activities in the links

\( T \) : link

\( \text{type} \) : synchronizer type of \( p \)

\( \text{count} \) : synchronous branches of \( p \)

\( \_\_\text{arc} \) : count of in-arcs

\( \_\_\text{needarc} \) : count of needed inc-arcs

\( \_\_\text{arrived} \) : count of inc-arcs arrived

\( \wp \) : business flow form

Logic algorithm:

If \( t \) is over, then

Step1: Compute the post set \( p \)

\( p = \text{get \_ post \_ set}(t) \)

Step2: Acquire the parameters of synchronizer

If \( p \) is end place, //end place

End

Else

\( \text{type} = \text{fun \_ gettype}(p) \);

\( \text{count} = \text{fun \_ getcount}(p) \);

\( \_\_\text{arc} = \text{fun \_ getin \_ arc}(p) \);

\( \_\_\text{arrived} = \text{fun \_ getin \_ arrived}(p) \);

Step3: Synchronous behavior begins

{ Case “Full \_ SYN”

If \( \_\_\text{arrived} = \_\_\text{needarc} \)

\( \wp = \text{merge}(p) \) /*merge all the needed synchronous branches*/

\( T = \text{Compute \_ next \_ activity}(p) \) /*compute the needed branch for next link*/

\( \text{Activate}(T) \) /*activate the branch*/

Else

\( \text{Wait}(p) \) /*wait for synchronization of other branches*/

Endif

Case “Non \_ SYN”

\( T = \text{compute \_ next \_ activity}(p) \)

\( \text{Activate}(T) \)

If \( \_\_\text{arrived}(p) < \_\_\text{needarc}(p) \) /*there are still activated in-arcs to be finished*/

\( \text{copy \_ activity}(T) \) /*copy a case for activity T*/

End if

Case “Compet \_ SYN”

If \( \_\_\text{arrived}(p) = \text{count}(p) \)

\( \wp = \text{merge}(p) \) /*merge all the forms of synchronous branches*/

\( T = \text{compute \_ next \_ activity}(p) \)

\( \text{Activate}(T) \)

Else if \( \_\_\text{arrived}(p) < \text{count}(p) \)

\( \text{Wait}(p) \)

End if

} //end switch

IV. IMPLEMENTATION ANALYSIS

Figure 7 provides a general cases figure of a hotel management system with top-level dataflow. It can be seen from this figure there are 4 types of users in this system. The connection lines between users and systems indicate the direction of dataflow. The main roles in this system include hotel reception, hotel members, hotel masters and system managers. The key functions are housekeeping management, occupancy management, room reservation, check-out, integrated query statistics, member management, and basic data maintenance etc. The hotel reception can involve in some operations including housekeeping management, occupancy management, member management and reservation as well as check-out [17]. The corresponding operation authorizations of other users’ types are also indicated in this figure.

A. Case of Non Synchronous Mode

In order to improve the operation efficiency of the company, when clients are booking rooms, the hotel reception only submits users’ registration information to room management department and room management department will distribute corresponding rooms according to clients’ requests. In addition, room management department can send several rooms which meet the requests for the reception to select. If clients have booked one room in advance, the following conditions should be satisfied:

1) Reservation in advance and pay attention to rooms’ status. Other tenants can occupy rooms before reserved dates. After online members submit the reservation information, system will set up “reserved” status on room status in database.

2) Non-occupancy treatment on due reserved room. If reserved room is due at reserved time and the room is still not used, “idle” state of homing will be set up.
(3) Members check-out. For the online members’ room reserved, before reserved time is invalid, the clients can cancel operations of hotel rooms’ reservation. After submitting canceling systematic operations, the system will set up “idle” state in database of room state.

During this business, we direct model based on original synchronizer and the following model is acquired:

- A: Submitting rooms reservation information to room management department;
- B: Rooms distribution;
- C: Members’ reservation;
- D: Members check-out;

\[ P_1 = (T_1, T_2, (1, 2), (Full\_SYN,*)) \] denotes complete synchronization mode;
\[ P_2 = (T_1, T_2, (2, 1), (Non\_SYN,1)) \] denotes non synchronization.

It can be found that during the business process of non-synchronizer mode on original synchronizer modeling, direct modeling cannot be set up and the mode structure must be changed for modeling. However, the extended synchronizer can set up business process of non-synchronizer pattern directly.

### B. Cases of Competitive Synchronization

In large cities the clients’ feedback system in most luxury hotels has a pattern of network application as “headquarter-branch maintenance” [18]. Under this mode, there is a client of branch-company management involving several branch companies areas. If one client administered from one branch-company has complaints, the investigation group from Customer Service Center Company will be sent to branch-companies in this area. For example, there are three branch-companies as A, B and C. Since this client is only probably in one area under this branch-company and any one branch-company completes, the system will return statement information without waiting for treatment results form the other two branch-companies.

\[ P_1 = (T_1, T_2, (1,2), (1,2), (Full\_SYN,*)) \] denotes complete synchronization mode;
\[ P_2 = (T_1, T_2, (2,1), (2,1), (Non\_SYN,1)) \] denotes non synchronization.

Figure 10. Complaint handling process
Based on above businesses, if the synchronizer is modeled to be sent to three branch-companies at the same time, the process will continue until three branches' completeness. However, in practice, it can only be dealt with one branch-company. In this way, it will result in unlimited waiting of this business at point \( P_3 \). Its model is shown as figure 11.

Figure 11. Process modeling by original synchronizer

\[
\begin{align*}
P_1 : P_1 &= (T_1, T_2(1,3), (\text{Full SYN}, *)) \quad \text{complete synchronization mode;} \\
P_2 : P_2 &= (T_1, T_3(3,1), (\text{Comp SYN}, 1)) \quad \text{competitive synchronization mode:}
\end{align*}
\]

The meanings of each place and transition are shown as:

- \( A \), \( B \), \( C \), \( D \) are the same to the above.

V. CONCLUSION

The difference between layered theory of workflow process model and WF-net originates from understanding and applied to Petri Net theory. Layered theory of workflow process model is accomplished under the guidance of synchronism from general net theory. The synchronizer is the conceptual foundation of these models: On one hand, the synchronizer connects all tasks in business process into an entirety and displays the whole process structure; on the other hand, synchronizer controls task operation to guarantee dynamic synchronization between tasks. At present, layered theory of workflow process model based on Petri Net synchronizer has become the most effective tool in Petri nets modeling method. For the problems in modeling of logic layer in hierarchical workflow process model, the paper proposes an extended Petri Net synchronizer and it summarizes currently general workflow model into three synchronization patterns: complete synchronization, competitive synchronization and non synchronization. It effectively solves the problems that involve the multi-combination and discriminator cannot adopt synchronizer modeling. Compared to previous synchronizers, it has improved the abilities of modeling discriminator and multi-combination. Meanwhile, it provides specific realization algorithm of this extended synchronizer. We also prove the modeling ability of improved synchronizer with the cases of hotel information management system.

REFERENCES


