DESCRIPTION OF WORKFLOW PATTERNS BASED ON P/T NETS

Guofu Zhou
School of Computer, Wuhan University
Wuhan University, Wuhan, 430072, China

Yanxiang He
Zhoumin Du

Keywords: Workflow Patterns, Workflow process, P/T nets.

Abstract: Through comparing and analyzing Aalst's workflow patterns, we model these patterns with P/T nets without additional elements. Based on these models, the number of Aalst's patterns can be reduced significantly. Moreover, the synchronic distance is also presented to specify workflow patterns.

1 INTRODUCTION

The theoretical foundation of workflow has become a hot problem. And, Petri nets is famous for the feature of describing the concurrent semantics with rich analysis techniques (CY. Yuan, 1998). Therefore, Petri nets is an ideal modelling tool of workflow process (WfMC, 1995). In (Aalst, 1998; Aalst et al., 2000; Aalst, 2002), Aalst presents four special kinds of transitions, four triggers and twenty workflow patterns. Although there are advantages in describing the semantics of workflow process by such new elements, one obvious disadvantage is there are too many additional elements to ensure conciseness. Moreover, not all workflow patterns provided by Aalst are necessary. In this paper, we model Aalst’s workflow patterns with P/T nets, and conclude not all Aalst’s workflow patterns are necessary. Additionally, an algebra method is presented to specify patterns.

2 MODELLING PATTERNS

In workflow process, workpiece, wp for short, is manipulated by activities. An activity is an operation task performed by one role on wp. Connector connects successive activities and controls the flow direction of workflow process. Connector are called workflow pattern or pattern.

What we should pay much attention to logic and schedule rule. Logic is the framework of workflow process and it will not change in all workflow instances. Schedule rule is effected by instance data of workflow process and is included in wp. Abstracting logic from workflow process is a key step to model pattern with P/T nets. In this paper, we actually discuss how to model logic of patterns. Aalst’s pattern (Aalst, 1999) specification can be simplified if the logic is separated from scheduling rules.

In this paper, connector is represented by P-element (a circle or a broken line circle or ellipse). Activity is represented by T-element (a rectangle or a broken line rectangle). The relationship between activity and connector is represented by the directed arc.

SEQUENCE is modelled as Fig.1. Where, activity A does not produce any control data. Control data is to regulate the flow direction of workflow process. For example, a role writes a sentence in wp: "please sent it to Mary". Where, the sentence "sent it to Mary" is control data. For convenience, A is called input activity and B is called output activity. AND-Split is modelled as Fig.2. Where, activity A does not produce any control data. The broken line ellipse denotes AND-Split. wp is replicated to several copies in workflow management system, which flow to the output activities, e.g. one copy to B and another...
Figure 1: SEQUENCE.

one to C. **AND-Join** is modelled as Fig.3. Where, $A_1 \cdots A_n$ don’t produce any *control data*. Through AND-Join, workflow management system integrates several copies of $wp$ into one $wp$. **XOR-Split** is modelled as Fig.4 or Fig.5. In Fig.4, activity $A$ does not produce any *control data*. Based on $wp$ and *related data*, XOR-Split chooses one activity from $B_1 \cdots B_n$ to activate. So $A$ determines which one of $B_1 \cdots B_n$ to be activated. Specially, XOR-Split has another representation (Fig.5). Where, activity $A$ contains exclusive subactivities $A_1 \cdots A_n$. Each $A_i$ produces *control data*. Therefore, A determines to activate which one of $A_1 \cdots A_n$. Because $A_1 \cdots A_n$ are exclusive, $A_i$ is activated implies $B_j$ will not be activated ($1 \leq i \neq j \leq n$). **XOR-Join** is modelled as Fig.6. Where, $S_1$ and $S_2$ form a *connector*. The token in $S_1$ determines only one of $A_1 \cdots A_n$ can be activated and it will flow into $S_2$. $C$ can be activated only if $S_2$ has one token. **OR-Split** is modelled as Fig.7. Where, $B_i$ denotes that activity $B_i$($i = 1 \cdots n$) is not be activated. When design *workflow process*, we can’t know which activity will be activated. So all possible activities must be listed, and each activity will either be activated or not. The option will be determined by workflow management system based on *related data* and *control data* produced by $A$. Actually, $B_i$ can be regarded as an error output. **OR-Join** is modelled as Fig.8 or Fig.9. In Fig.8, $x$ in $S_1$ denotes that $S_1$ has $x$ tokens, and the number of tokens denotes the number of activities can be activated. Where, $x \leq n$. Each $A_i$($i = 1 \cdots n$) can be activated once and produce one token for $S_2$. $x$ in the directed arc from $S_2$ to $C$ denotes $C$ can be activated only if $S_2$ has $x$ tokens. Workflow management system chooses activities from $A_1 \cdots A_n$ to activate based on $wp$ and *related data*. OR-Join has another representation (Fig.9). Where, $A_1 \cdots A_n$ can determine whether activate $B_i$ or not. $B_i$ denotes $B_i$ is not activated and it can be regarded as an error output. **Multiple-Merge** is modelled as Fig.10. Where, $x$ tokens in $S_1$ denotes that $x$ activities of $A_1 \cdots A_n$ can be activated concurrently. The weight of directed arc from $S_2$ to $C$ is 1. If $S_2$ has one token, $C$ will be activated once. Each $A_i$($i = 1 \cdots n$) can produce
be activated and consume the tokens of both $S_2$. $x$ tokens in $S_2$ implies $C$ will be activated $x$ times. **Discriminator** is modelled as Fig.11. Where, when $S_1$ and $S_2$ have one token respectively, $D$ can be activated and consume the tokens of both $S_2$ and $S_1$. Although each $B_i (i = 1 \ldots m)$ can produce one token for $S_2$, $D$ can’t be activated again because $S_1$ has no token. Actually, Discriminator is a special case of N-out-of-M Join. In Fig.12, the weight of directed arc from $S_2$ to $D$ is $n$ and the number of element of $S_2$’s preset is $m (n \leq m)$. When $S_2$ has $n$ tokens and $S_1$ has one token, $D$ can be activated. In Discriminator, $n = 1$. In(Aalst et al., 2000), Aalst also mentions some other patterns. **Arbitrary Cycles** only modifies $w_p$’s content but doesn’t change the logic of workflow process. So it can be regard as a loop. **Implicit Termination** denotes the process is stop. Obviously it is not necessary to study specially. **Multiple Instance Without Synchronization**, **Multiple Instances With a Priori Design Time Knowledge**, **Multiple Instances With a Priori Run Time Knowledge**, and **Multiple Instances Without a Priori Run Time Knowledge** are related with multiple instances, therefore they should be discussed as process instances. **Deferred Choice** is an error of resources if time is regarded as a kind of resource. So it shall not also be considered as a pattern. **Interleaved Parallel Routing** determine the activating order of the concurrent activities, so this pattern is similar essentially to **SEQUENCE**. **Milestone** can activate a new activity without terminating itself, so actually **Milestone** is a activity. **Cancel Activity** and **Cancel Case** are also not patterns but error outputs.

### 3 Pattern Specification

**XOR-Split** is a special case of **OR-Split** if only one of output activities can be activated. In **XOR-Join**, only input activity is activated and output activity will be activated only once. In **OR-Join**, some of input activities are activated and output activity will be activated only once. In **Multiple Merge**, some of input activities are activated and output activity will be activated corresponding times. Therefore, the three patterns are the special cases of **merge**. In Fig.13, when $m = n = 1$, merge is **XOR-Join**. When $m = n > 1$, merge is **OR-Join**. When $m > n = 1$, merge is **Multiple Merge**. When $n = 1$, **N-out-of-M** is **Discriminator**. Therefore, only **SEQUENCE**, **AND-Split**, **AND-Join**, **OR-Split**, **OR-Join** and **N-out-of-M** merge are necessary.

3.1 Specification

We specify the patterns by the algebra of Petri nets. Let $A$ be the set of activities and $E_1, E_2$ be subsets of $A$. $E_1 \subseteq E_2$ denotes $E_1$’s activities are activated before $E_2$’s, and $E_1 \subseteq E_2$ denotes $E_1 \cap E_2 \neq \emptyset \land E_1 - E_2 < E_2 - E_1$. It is called a point that $w_p$ (and its copies) is in the hand of a role. Let $p_1, p_2$ be arbitrary points. $\#(E_i, p_1, p_2)$ denotes the number of occurrences of $E_i$’s activities from $p_1$ to $p_2 (i = 0, 1, 2)$. $p_0$ is the point before the start activity.

**Definition 1** **Synchronous Distance** $\sigma(E_1, E_2)$, the synchronous distance between $E_1$ and $E_2$, is defined as below:

$$
\sigma(E_1, E_2) = \begin{cases} 
\max_{p_1, p_2 \in P} \left\{ \left\lvert \#(E_1, p_1, p_2) - \#(E_2, p_1, p_2) \right\rvert \right\} & \text{if } \#E_1 = \#E_2 \\
\varepsilon & \text{if } \#E_1 \neq \#E_2
\end{cases}
$$

Where, $P$ is the set of all points, and $\#E_i = \max_{p \in P} \left\{ \#(E_i, p_0, p) \right\} (i = 1, 2)$. $\varepsilon$ denotes no value.
Definition 2 Asymmetric Synchronous Distance

\[ \bar{\sigma}(E_1, E_2) \]

(\[E_1 \] to \(E_2\)), the asymmetric synchronous distance from \(E_1\) to \(E_2\), is defined as below:

\[ \bar{\sigma}(E_1, E_2) = \begin{cases} \sigma(E_1, E_2) & \text{if } E_1 \leq E_2 \\ \varepsilon & \text{if } E_1 > E_2 \end{cases} \]

Based on the above definition, we can easily get specifications: SEQUENCE(Fig.1) is specified by \(\bar{\sigma}(A, B) = 1\). AND-Split(Fig.2) is specified by \(\bar{\sigma}(A, B) = \bar{\sigma}(A, C) = 1\). AND-Join(Fig.3) is specified respectively by \(\bar{\sigma}(A_1, B) = \bar{\sigma}(A_n, C) = 1\) and \(\bar{\sigma}(A_1, A_1) = \varepsilon\).

Let \(E_3, E_4\) be multiple sets on \(A\), i.e. \(E_j : A \rightarrow \{0, 1, \ldots\}\). \((j = 3, 4)\) is a mapping from \(A\) to the set of non-negative integers. For example, \(a\) is an element of \(A\) and \(E_j(a)\) is the occurrence number of \(a\) in \(E_j\). To count the weighted occurrences of \(E_j\)'s activities, we have definitions respectively: \(\sigma(E_j, p) = \sum_{a \in E_j} \#(a, p)\) and \(\sigma_j = \max\{\sigma(E_j, p)\}\).

Where, \(\#(a, p) = \#(a, p_0, p)\), \(E_j(a)\) is the occurrence number of \(a\) in \(E_j\).

Definition 3 Weighted Synchronous Distance

\[ \sigma(E_3, E_4) = \begin{cases} \max_{p \in P}\{\sigma(E_3, p) - \sigma(E_4, p)\} & \text{if } E_3 = E_4' \\ \varepsilon & \text{if } E_3 \neq E_4 \end{cases} \]

Definition 4 Weighted Asymmetric Synchronous Distance

\[ \bar{\sigma}(E_3, E_4) = \begin{cases} \sigma(E_3, E_4) & \text{if } E_3 \leq E_4' \\ \varepsilon & \text{if } E_3 > E_4' \end{cases} \]

Where \(E_j' = \{a | a \in A \wedge E_j(a) \neq 0\}, j = 3, 4\). For conciseness, there are:

\[ \bar{\sigma}(E_3, E_4) = \begin{cases} \sigma(a, E_3) & \text{if } E_3 = \{a\} \\ \sigma(a, E_4) & \text{if } E_4 = \{b\} \\ \sigma(a, b) & \text{if } E_3 = \{a\} \text{ and } E_4 = \{b\} \end{cases} \]

For example, the pattern in Fig.14 can be specified

![Figure 14: Pattern OR.](image-url)

by \(\exists n : \bar{\sigma}(n \bullet a, \{b_1, b_2, \ldots, b_m\}) = n \wedge \sigma(b_i, b_j) = \varepsilon(i \neq j)\) and \(\exists n : \bar{\sigma}(\{a_1, a_2, \ldots, a_m\}, n \bullet b) = n \wedge \sigma(a_i, a_j) = \varepsilon(i \neq j)\). Where, \(0 < n \leq m\), \(n \bullet a\) denotes the number of \(a\), and \(n\) is called OR-number. When \(n = 1\), the pattern is XOR (exclusive OR). OR-number may be given at any time. The specification of pattern OR implies synchronization, including synchronized multi-choice and synchronized multi-merge. If not requiring synchronization, i.e., each \(a_i(i = 1, \ldots, m)\) will activate \(b\) once, pattern OR can be specified by \(\exists n : \bar{\sigma}(\{a_1, a_2, \ldots, a_m\}, b) = n\) where \(0 < n \leq m\). This OR-number denotes the number of times \(b\) will be executed.

From above discussion, similar to the graphic representation of patterns, Synchronous distance with its extensions also captures logic of workflow process.

4 CONCLUSION

We have analyzed and modelled all Aalst’s patterns with P/T nets. Some patterns shall not be regarded as patterns. The significant thing is we model the patterns without any extensions to Petri nets or triggers. Furthermore, we propose synchronous distance to specify logic of workflow process, which can not only achieve the same effect as Aalst’s but also be more concise and reasonable. By our standpoint, the model of workflow process can be divided into two layers: control flow layer and data flow layer. Control flow layer (modelled by Petri nets) is to check the validation and feasibility of workflow process. Data flow layer (data flow) will be described and simulated by UniNet (GF.Zhou, 2003).

REFERENCES


