Dynamically Analyzing Time Constraints in Workflow Systems with Fixed-Date Constraint

Rui Han¹, Yingbo Liu², Lijie Wen³, Jianmin Wang⁴

¹School of Software, Tsinghua University, Beijing,
²Key Laboratory for Information System Security, Ministry of Education,
³Tsinghua National Laboratory for Informativasion Science and Technology (TNList), P.R. China, 100084
¹,³{(hanr07, lyb01, wenlj00)@mails.tsinghua.edu.cn, ³jimwang@tsinghua.edu.cn

Abstract—In workflow management systems (WFMSs), time management plays an essential role in controlling the lifecycle of business processes. Especially, run-time analysis of time constraints is necessary to help process manager proactively detect possible deadline violations and appropriately handle these violations. Traditional time constraint analyses either present deterministic results which are too restrictive in highly uncertain workflow processes, or only consider static analysis at workflow build-time. For such an issue, this paper proposes a dynamic approach for analyzing time constraints during process execution. To be specific, based on a Petri-net-extended stochastic model, this approach first analyzes activity instances’ continuous probabilities of satisfying time constraints when a process instance is initiated. Afterwards, during the execution of this process instance, the approach dynamically updates these probabilities whenever an activity instance is completed. Moreover, an example process instance in real-world WFMSs shows the practicality of our approach.

I. INTRODUCTION

Workflow management systems (WFMSs) provide sophisticated modeling tools for controlling, monitoring, optimizing and supporting business processes [1]. Hence WFMSs should provide time management as an important function to control the lifecycle of business processes in a timely manner. In particular, during the execution of business processes, pro-active mechanisms are needed to notify process managers about potential deadline violations and help them take necessary steps to avoid these time failures [2].

Generally, time constraints include three types: lower bound constraint, upper bound constraint, and fixed-date constraint [2-5]. A lower bound constraint is a relative time value between two activities that the duration between them must be less than or equal to it. An upper bound constraint is symmetrical to a lower bound constraint. A fixed-date constraint denotes the latest time that an activity instance (also called a work item or task) must be completed with respect to the start time of the process. The procedure of verifying that the time constraints imposed on WFMSs are satisfied or violated is called time constraint analysis.

Most of conventional time constraint analyses give deterministic analysis results such as time constraints are satisfied or violated [2-9]. At workflow build-time, some other research focuses on setting fixed-date constraints [10], or statically analyzing time constraints without considering run-time issues such as fixed-date constraints or completion of activity instances [11, 12]. However, process instances execute in highly dynamic workflow environment. First, activity instances’ (execution) durations are uncertain and execution paths are unpredictable. Second, in a process instance, different activity instances are allocated different fixed-date constraints. Also, in different process instances, instances of a same activity are assigned different fixed-date constraints. These uncertainties cause activity instances’ various possibilities of satisfying time constraints. Furthermore, when a process instance is executing, these possibilities of uncompleted activity instances are affected by the already-executed portion of the process instance. Consequently, at workflow run-time, traditional work lacks in analyzing time constraints in a probabilistic way as well as real-time updating of the analysis result.

This paper builds upon previous work on static time constraint analysis at workflow build-time. Here we extend the traditional PTCWF-nets (Probabilistic Time Constraint WorkFlow Nets) [11, 12] to include fixed-date constraint, thus define the extended PTCWF-nets (i.e., \( \text{PTCWF-nets} \)) to model the time-constrained process instance during workflow execution. Based on \( \text{PTCWF-nets} \), the paper proposes a two-stage approach as a means of quantitatively and dynamically analyzing time constraints. Specifically, the approach first pre-analyzes every activity instance’s continuous probability of satisfying time constraints at the first stage. This probability is denoted as successful execution ratio. Afterwards, during the execution of the process instance, the approach dynamically updates successful execution ratios of uncompleted activity instances when some activity instances are completed. This stochastic analysis result can help process managers accurately monitor temporal status of the executing process instance. Therefore, process managers can proactively identify possible time constraint violations and properly handle these violations.

The rest of the paper is organized as follows: we begin by introducing problem analysis and defining the \( \text{PTCWF-nets} \) (Section II). Given this necessary information, we describe our approach for dynamically analyzing time constraints (Section III) and verify the effectiveness of our approach in Tsinghua InfoTech Product Lifecycle Management (TiPLM)
II. PROBLEM ANALYSIS AND PTCWF-NETS

In this section, we first propose a motivating example as a means of analyzing problems in run-time time constraint analysis. Then, we formally define time constraints and the PTCWF-nets.

A. Motivating Example and Problem Analysis

The motivating example is implemented in the TiPLM model editor. In TiPLM model editor, a workflow process is described by a directed graph that has four different kinds of nodes: activity nodes, route nodes, start node, and end node. Nodes are connected by directed links to define different kinds of control flows [13]. Fig. 1 shows a screenshot of the TiPLM workflow model editor, in which a process of altering contract is being edited.

A new requirement of altering contract can initiate one instance of the process in Fig. 1. First, a contract manager handles the contract for alteration (activity instance Manage_contract). Then, three different departments separately countersign the contract (activity instance countersign1 to countersign3). Next, the contract manager reviews the materials and decides whether they are up to the standard (activity instance Review). If the proposal is approved (activity instance Approve), two operations related to the alteration are conducted (activity instance Maintain_product and Manage finance), and the process proceeds to release for downstream organizations (activity instance Release). Otherwise (activity instance Disapprove), a link is defined to redirect the execution from the ‘Review’ node back to the ‘Manage_contract’ node, so that the manager should prepare the alteration again.

![Fig. 1 Example process of altering contract in TiPLM workflow model editor](image)

Here, we take activity instance Maintain_product in Fig. 1’s example process to explain three problems in dynamic analysis of time constraints during process execution.

1) Three kinds of time constraints are criteria of successful execution. Activity instance Maintain_product should satisfy three time constraints: when its predecessor activity instance Approve is completed, Maintain_product has to wait 3.17 hours before being able to start (lower bound constraint) and should be finished within 17.24 hours (upper bound constraint). Also, when the process instance starts, Maintain_product must be completed within 240 hours (fixed-date constraint).

2) Two uncertainties in process instances decide the analysis result. Due to the dynamic workflow environment, activity instance Maintain_product’s duration ranges between 11.80 to 14.68 hours (named uncertain duration parameter). In addition, when activity instance Approve is completed, there is a 36% of selecting Maintain_product for execution (named uncertain path parameter). These two uncertain parameters cause Maintain_product’s successful execution ratio to be 97.25%.

3) Completed activity instances influence the analysis results. If activity instance Approve is completed later than expectation, its successor activity instance Maintain_product’s start time and completion time are delayed. Hence Maintain_product’s successful execution ratio drops to 90.02%.

To conclude, the basic requirements for run-time time constraint analysis can be simply put as provides quantitative analysis results by comprehensively considering three time constraints and two uncertainties, as well as dynamically updating these analysis results. However, to the best of our knowledge, little efforts have been dedicated to dynamically analyze time constraints in a probabilistic way.

B. Time Constraints and Extended PTCWF-nets

Petri net (PN) is a widely selected mathematical foundation for the analysis of WFMSs [1]. A classical PN is a triple \((P, T, F)\): \(P\) is a finite set of places, \(T\) is a finite set of transitions \((P \cap T=\emptyset)\), and \(F \subseteq (P \times T) \cup (T \times P)\) is a set of arcs (flow relation). Circles represent places and rectangles represent transitions. The nodes are connected via directed arcs and connections between two nodes of the same type are not allowed [1]. A PN is called as a Workflow net (WF-net) satisfying the following two requirements: 1) PN has two special places: \(i\) and \(o\). Place \(i\) is the source place: \(\bullet i = \emptyset\). Place \(o\) is the sink place: \(\bullet o = \emptyset\); 2) If we add a transition \(t’\) to PN which connects place \(o\) with \(i\) (i.e., \(t’ \subseteq \{o\}\) and \(t’ \subseteq \{i\}\)), then the resulting PN is strongly connected.

![Diagram](image)
Inheriting and developing from classical WF-nets [1], PTCWF-nets [11, 12] define the ways of mapping lower and upper bound constraints to set of places $P$, uncertain duration parameters to set of transitions $T$, and uncertain path parameters to set of arcs $F$, respectively. In addition, we define PTCWF-nets by mapping fixed-date constraints to PTCWF-nets’ set of transitions $T$. In PTCWF-nets, transitions represent activity instances.

**Definition 1**: (Place constraint function $C_p$). Let $(P, T, F)$ be a WF-net, $C_p : P \rightarrow T \times T$ is an assignment of time intervals to places. For a place $p \in P$, $C_p(p)$ captures the time interval during which a token in $p$ is available for its output transitions. We define $p.min$ and $p.max$ for each place $p \in P$ in such a way that $C_p(p) = (p.min, p.max)$.

For a place $p$, $p.min$ and $p.max$ respectively describe the minimum and maximum time interval between the preceding and succeeding activity instances.

**Definition 2**: (Transition constraint function $C_t$). Let $(P, T, F)$ be a WF-net, $C_t : T \rightarrow R$ is an assignment of fixed-dates to transitions.

For a transition $t$, $C_t(t)$ represents an activity instance’s fixed-date constraint.

**Definition 3**: (Transition parameter function $B_t$). Let $(P, T, F)$ be a WF-net, $B_t : T \rightarrow R \times R$ is an assignment of durations to transitions. For a transition $t \in T$, $B_t(t) = (\mu, \sigma)$ is governed by a normal distribution, i.e., $B_t(t) \sim N(\mu, \sigma)$.

For generally, all activity durations are assumed to follow the normal distribution model $N(\mu, \sigma)$ [14, 15].

**Definition 4**: (Arc parameter function $B_e$). Let $(P, T, F)$ be a WF-net, $B_e : F \rightarrow R$ is an assignment of probability values to arcs. For any place $p \in P$, $\sum_{f \in \Delta(p)} B_e(f) = 1$.

For a transition $t$, $B_e(f)$ represents the probability of selecting execution paths in the process instance.

**Definition 5**: (PTCWF-nets). A PTCWF-net is a six-tuple $\text{PTCWF-net} = (P, T, F; C_p, B_t, B_e)$, where $(P, T, F)$ is a classical WF-net.

**Definition 6**: (PTCWF-net). An extended PTCWF-net is a seven-tuple $\text{PTCWF-net}_e = (P, T, F; C_p, C_t, B_t, B_e)$, where $(P, T, F)$ is a classical WF-net.

PTCWF-nets have the property of safe and free-choice [1]. Also, PTCWF-nets are isomorphic to continuous time Markov chains [16]. As shown in Table 1, PTCWF-nets include four types of workflow control structures: sequential, parallel, selective, and iterative. These four structures can be combined to model majority of typical business processes in various domains, such as manufacture factory, hospital, bank, and so on [1].

## III. A Two-Stage Approach for Dynamically Analyzing Time Constraints

In this section, we respectively explain stage 1 and stage 2 of our approach in Section III-C and Section III-D, as shown in TABLE II. This approach is conducted based on the structural verification [17], which means that the process instance considered in the approach is free of structural conflicts (deadlock).

### TABLE II

<table>
<thead>
<tr>
<th>Overview</th>
<th>Method</th>
<th>Input: Process instance, workflow event log.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1: Pre-analysis</td>
<td>Two-stage approach for dynamically analyzing time constraints.</td>
<td>Pre-analyzing every activity instance’s successful execution ratio.</td>
</tr>
<tr>
<td>Stage 2: Dynamic updating</td>
<td>Activity instances’ real-time successful execution ratios</td>
<td>Dynamically updating uncompleted activity instances’ successful execution ratios</td>
</tr>
</tbody>
</table>

### C. Pre-analysis of Every Activity Instance’s Successful Execution Ratio

At run-time initialization stage, when the business process is initiated, time constraints (definition 1 and 2) and parameters (definition 3 and 4) are assigned to this process instance. Then, the approach models the process instance into a PTCWF-net and pre-analyzes every transition’s successful firing ratio to represent activity instance’s successful execution ratio.

As shown in Fig.2, for a transition $t$ in the PTCWF-net, given that $t$’s duration $B_t(t) \sim N(\mu, \sigma)$, its completion time is $(s^t + \epsilon_{min}), s^t + \epsilon_{max}$, and its enabled time interval is $i^t = (\epsilon_{min}, \epsilon_{max})$. The firing of $t$ is considered as successful only if $t$ completes before both its maximum enabled time $\epsilon_{max}$ (i.e., relative deadline) and its fixed-date

---

**TABLE I**

<table>
<thead>
<tr>
<th>Workflow control structure</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td><img src="image" alt="Sequential" /></td>
</tr>
<tr>
<td>Parallel</td>
<td><img src="image" alt="Parallel" /></td>
</tr>
<tr>
<td>Selective</td>
<td><img src="image" alt="Selective" /></td>
</tr>
<tr>
<td>Iterative</td>
<td><img src="image" alt="Iterative" /></td>
</tr>
</tbody>
</table>
$$\text{constraint } C_i(t) \text{ (i.e., absolute deadline), namely, } \\
(s^t + B_i(t)) \in i_j = (s^j + d_{\text{min}}^j, \text{Min } \{ e_{\text{max}}, C_i(t) \}). \text{ Therefore, } t_i \text{’s } \\
\text{successful firing ratio is the probability integral of random variable } (s^t + B_i(t)) \text{ over the interval of } i_j.$$

**Definition 7:** (Transition’s successful firing ratio). For a transition $t$ in PTCWF-nets, its successful firing ratio: $r^P = P(s^t + d_{\text{min}}^t < s^t + B_i(t) < \text{Min } \{ e_{\text{max}}, C_i(t) \})$

i.e., $r^P = P(d_{\text{min}}^t < B_i(t) < \text{Min } \{ e_{\text{max}}, C_i(t) \} - s^t)$ (1)

The above pre-analysis is based on the static analysis at build-time stage. As proposed in [12], through applying “3m” rule [17], the approach gets transition $t_i$’s minimum duration: $d^t_{\text{min}} = \mu_t - 3\sigma_t$, and the maximum duration: $d^t_{\text{max}} = \mu_t + 3\sigma_t$. In addition, transition $t$ is enabled earliest when all its input places has available tokens: $e_{\text{min}}^t = \text{Max } \{ a^t + p_{\text{min}} | p \in \bullet t \}$ and latest when any of its input places has no available token: $e_{\text{max}}^t = \text{Min } \{ a^t + p_{\text{max}} | p \in \bullet t \}$. Also, $t$ is assumed to start as soon as it is enabled: $s^t = e_{\text{min}}^t$. Moreover, $e_{\text{min}}^t$, $e_{\text{max}}^t$, and $s^t$ are all related to token’s earliest available time in place $p$ ($p \in \bullet t$): $d^p$. We take place $p_2$ in TABLE 1’s four types of workflow control structures to explain the calculation of token’s earliest available time [12].

–For a sequential or parallel control structure, $a^{p_2}$ is transition $t_1$’s expected completion time: $a^{p_2} = p_{\text{max}} + \mu_{t_1}$.

–For a selective control structure, $a^{p_2}$ is the probability synthesis of transition $t_1$ and $t_2$’s expected completion time: $B_2(f_2) \times (a^{p_2} + p_{\text{min}} + \mu_{t_1}) + B_1(f_1) \times (a^{p_1} + p_{\text{min}} + \mu_{t_1})$.

–For an iterative control structure, $a^{p_2}$ is different for $p_2$’s output transition $t_2$, which is inside the loop body and $t_2$ which is outside the loop body. For $t_2$, $a^{p_2}$ is $t_2$’s first expected completion time: $a^{p_2} = p_{\text{min}} + \mu_{t_2}$. For $t_2$, $a^{p_2}$ is the time when $t_2$ and $t_2$’s iterative firings complete: $a^{p_2} + p_{\text{min}} + \mu_{t_2}$ \times (B_2(f_2)/B_2(f_2+1)) \times p_{\text{min}} + \mu_{t_2}$ $\times (B_2(f_2)/B_2(f_2+1))$.

Fig. 3 shows the pseudocode of the whole pre-analysis algorithm. This algorithm’s time complexity is $O(n + m + k)$ given that in the PTCWF-net, set $T$ has $n$ transitions, set $P$ has $m$ places, set $F$ has $k$ arcs $(n, m, k \geq 1)$, and the time complexity of basic breadth-first traversal (BFT) strategy is $O(n + m + k)$ for directed graph PTCWF-net [18].

In Fig. 3, the algorithm utilizes the BFT strategy [18] to search the PTCWF-net from the source place $i$ to the sink place $o$ (line 1 and line 5). Every time the approach gets a new transition by BFT search, the approach pre-analyzes its successful firing ratio (line 3) and adds it the end of list $U$ (line 4). When the overall algorithm is completed, list $U$ contains all $n$ transitions and this list provides a basis for the dynamically updating at the next dynamic updating stage.

**Input:** The PTCWF-net, empty list $U$

**Output:** List $U$ with all transitions in the PTCWF-net and their pre-analyzed successful firing ratios

**Method**

1. $t := T.getTransitionByBFT();$
   // obtain the first transition from the PTCWF-net
2. WHILE $t \neq \emptyset$ DO
   BEGIN
   3. $r^P := P(d^t_{\text{min}} < B_i(t) < \text{Min } \{ e_{\text{max}}, C_i(t) \} - s^t)$;
      // pre-analyze $t$’s successful firing ratio
   4. $U := U \cup \{ t \};$
      // add $t$ to the end of list $U$
   5. $t := T.getTransitionByBFT();$
      // obtain a new transition
   END

**Function getTransitionByBFT()**

return a new transition from the PTCWF-net through BFT strategy; return $\emptyset$ if no new transition exists in the PTCWF-net.

Fig. 4 Algorithm for pre-analyzing every transition’s successful firing ratio

**D. Dynamically Updating Uncompleted Activity Instances’ Successful Execution Ratios**

At run-time execution stage, the approach monitors the execution of the process instance. Each time an activity instance (e.g., $a_i$) is completed, the approach updates other uncompleted activity instances’ successful execution ratios. Note that in the approach, only activity instances that range after $a_i$ in the process instance are considered based on two assumptions: 1) only these activity instances’ start time are affected by $a_i$’s completion time and they can only start after $a_i$ is completed; 2) the approach performs immediately before new activity instances are executed because the approach takes a short time with respect to activity instances’ execution time.

As discussed above, in the PTCWF-net, when a transition (e.g., $t_i$) completes, only transition (e.g., $t_i$) satisfies the following two conditions is selected from list $U$ as qualified transition to update its successful firing ratio: 1) $i$ ranges after completed transition $t_i$ in list $U$; 2) $i$ is $t_i$’s reachable transition in the PTCWF-net.

Fig. 4 shows the pseudocode of the dynamic updating algorithm. The time complexity of this algorithm is $O(n(n + m + k))$ given that list $U$ initially has $n$ transitions and each transition has $m$ output places at most in the PTCWF-net $(n, m \geq 1)$.

In Fig. 4, the algorithm first finds the completed transition $t_i$ from list $U$ (line 1). Then, the algorithm recalculates $t_i$’s output places’ token’s earliest available time (line 2 and line 3) and removes $t_i$ from list $U$ (line 5) because this list is set to contain uncompleted transitions in the algorithm. Next, the algorithm selects a qualified transition $t$ from list $U$ (line 7 to line 9), recomputes its start time (line 10), and updates its successful firing ratio according to formula 1 in definition 7 (line 11), and its output places’ token’s earliest available time (line 12 and line 13). Finally, if $t$ is not the last transition in
list $U$ (line 6), the algorithm performs a new iteration to update a new qualified transition’s successful firing ratio.

Input: completed transition $t_j$, list $U$
Output: Uncompleted transitions’ updated successful firing ratios

Method
1. Find completed transition $t_j$ in list $U$;
2. FOR EVERY place $p$ (if $p \in T$) DO
   BEGIN
   3. $p$.recalculateTokenAvailableTime();
   // recalculate $p$’s token’s earliest available time
   END
4. $t := U$.getNextTransition($t_j$);
   // obtain $t_j$’s next transition in list $U$
5. $U := U \backslash \{t_j\}$;
   // remove completed transition $t_j$ from list $U$
6. WHILE $t \neq \emptyset$ DO
   BEGIN
   7. $t := U$.getNextTransition($t$);
   8. IF $t$ is not reachable THEN RETURN;
   END
8. $s' := e_{\min} = \text{Max}(a^p + p.min[p \in T])$;
   // recalculate $t$’s start time
9. $r' := P(d_{\min} - B_0(t) - \text{Min}(e_m, C(0)) - s')$;
   // update $t$’s successful firing ratio
10. FOR EVERY place $p$ (if $p \in T$) DO
    BEGIN
    11. $p$.recalculateTokenAvailableTime();
    END
12. $t := T$.getNextTransition($t$);
   END

Function getNextTransition($t$)
// return $t_j$’s next transition in list $U$; return $\emptyset$ if $t$ is the last transition in list $U$.

Function recalculateTokenAvailableTime()
// recalculate token’s earliest available time in the place.

Fig. 4 Algorithm for updating uncompleted transitions’ successful firing ratios

In Fig. 4’s algorithm, token’s earliest available time in place $p$ is recalculated differently in four workflow control structures (line 3 and line 13). Given that transition $t_j$ and $t_k$ complete and their actual durations are known (i.e., $d^{t_j}$ and $d^{t_k}$), we take place $p^2$ in TABLE II to explain the recalculation of token’s earliest available time:

- For a sequential or parallel control structure, $a^{p^j}$ is transition $t_j$’s actual completion time: $s^{t_j} + d^{t_j} = a^{p^j} + p^{\text{min}} + d^{t_j}$.

- For a selective control structure, $a^{p^j}$ is the earlier actual completion time of transition $t_j$ and $t_k$: min {$a^{p^j} + p^{\text{min}} + d^{t_j}$, $a^{p^k} + p^{\text{min}} + d^{t_k}$}.

- For an iterative control structure, $a^{p^j}$ is different for $p^j$’s output transition $t_j$ which is inside the loop body and $t_k$ which is outside the loop body. Given that transition $t_j$ fires $(c^{t_j} + 1)$ times and $t_k$’s holding time is $(p^{\text{min}} + d^{t_k}) (1 \leq k \leq c^{t_j} + 1)$ in the $k$th firing. $t_j$’s fires $c^{t_j}$ times and its holding time is $(p^{\text{min}} + d^{t_j}) (1 \leq j \leq c^{t_j})$ in the $j$th firing. For $t_j$, $a^{p^j}$ is $t_j$’s first actual completion time: $(a^{p^j} + p^{\text{min}} + d^{t_j})$. For $t_k$, $a^{p^k}$ is the time when the iterative firings of $t_j$ and $t_k$ complete: $a^{p_k} + \sum_{k=1}^{c^{t_j}+1} (p^{\text{min}} + d^{t_k}) + \sum_{j=1}^{c^{t_j}} (p^{\text{min}} + d^{t_j})$.

IV. EXAMPLE AND EVALUATION

In this section, we implement our work as a prototype system by revising the TiPLM workflow model editor and evaluate the effectiveness of our approach by further illustrating the motivating example introduced in Section II-A.

In the first step, we model Fig. 1’s example process instance of altering contrast into a PTCWF-net, as shown in Fig. 5.

The time constraints and parameters of this PTCWF-net are presented in TABLE III. In TABLE III, the time unit is hour (referring to transitions and places), and only five uncertain path parameters of arc $f_i$ to $f_j$ are given because other arcs’ time parameters are 1.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$C_p(\mu, \sigma^2)$</th>
<th>$t$</th>
<th>$B_{f_i}(t)$</th>
<th>$C_{f_j}(t)$</th>
<th>$f$</th>
<th>$B_{f_j}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>(1.02, 6.98)</td>
<td>$t_1$</td>
<td>(5.52, 0.18)</td>
<td>10</td>
<td>$f_1$</td>
<td>0.33</td>
</tr>
<tr>
<td>$p_2$</td>
<td>(5.11, 68.27)</td>
<td>$t_2$</td>
<td>(57.11, 1.97)</td>
<td>80</td>
<td>$f_2$</td>
<td>0.24</td>
</tr>
<tr>
<td>$p_3$</td>
<td>(1.47, 57.86)</td>
<td>$t_3$</td>
<td>(61.26, 2.06)</td>
<td>80</td>
<td>$f_3$</td>
<td>0.43</td>
</tr>
<tr>
<td>$p_4$</td>
<td>(2.74, 12.85)</td>
<td>$t_4$</td>
<td>(58.12, 2.02)</td>
<td>80</td>
<td>$f_3$</td>
<td>0.36</td>
</tr>
<tr>
<td>$p_5$</td>
<td>(3.17, 17.24)</td>
<td>$t_5$</td>
<td>(52.73, 1.92)</td>
<td>130</td>
<td>$f_3$</td>
<td>0.64</td>
</tr>
<tr>
<td>$p_6$</td>
<td>(3.24, 15.72)</td>
<td>$t_6$</td>
<td>(9.52, 0.24)</td>
<td>150</td>
<td>$f_0$</td>
<td>-</td>
</tr>
<tr>
<td>$p_7$</td>
<td>(1.12, 36.05)</td>
<td>$t_7$</td>
<td>(9.52, 0.32)</td>
<td>220</td>
<td>$f_0$</td>
<td>-</td>
</tr>
<tr>
<td>$p_8$</td>
<td>(1.72, 37.27)</td>
<td>$t_8$</td>
<td>(13.24, 0.48)</td>
<td>240</td>
<td>$f_0$</td>
<td>-</td>
</tr>
<tr>
<td>$p_9$</td>
<td>(33.49, 1.17)</td>
<td>$t_{10}$</td>
<td>(11.72, 0.41)</td>
<td>240</td>
<td>$f_0$</td>
<td>-</td>
</tr>
</tbody>
</table>

Based on Fig. 5’s PTCWF-net, we demonstrate two stage of our approach. At the first pre-analysis stage, the approach analyzes all activity instances’ successful execution ratios, as shown Fig. 6.a. Also, at the end of the pre-analysis, list $U$ contains 10 transitions whose sequence follows the same search sequence of the PTCWF-net by BFT strategy. Fig. 6.b shows a snapshot of the second dynamic updating stage, in which the last three activity instances (Maintain_product, Manage_finance, and Release) are uncompleted, hence list $U$ only contains three transitions (i.e., $t_6$, $t_8$, and $t_{10}$). Given that, the last completed activity instance Approve’s actual duration is 9.83 hours rather than the expected 9.52 hours, thus the approach updates successful execution ratios of three
uncompleted activity instance *Maintain product, Manage finance*, and *Release*: 90.02%, 86.30% and 77.81%.

In the example, the analysis result proactively notifies process managers about possible deadline violations for uncompleted activity instances. To avoid these violations, process managers should respectively handle these activity instances with different values of successful execution ratios:

1) Activity instance *Maintain product* with high value (90.02%) is considered as feasible and meeting system’s requirement of stability. Hence process managers maintain these activity instances for execution without triggering any exception handling.

2) Activity instance *Manage finance* with middle value (86.30%) is treated by simpler and more cost-saving exception handlings, such as skipping or delaying some less important activity instances.

3) Activity instance *Release* with low value (77.81%) is treated by complicated and high-cost exception handlings, such as modifying process structure.

Note that how exception handlings should be triggered is beyond the scope of this paper, because it is outside the scope of time constraint analysis and is often related to some other overall aspects of the whole WFMSs such as quality of service (QoS) [19].

In conclusion, the above demonstration of the analyzing process evidently shows that our approach is effective for dynamically analyzing time constraints, and it has solved the three problems in run-time analysis of time constraints as proposed in Section II–A.

V. RELATED WORK

In this section, we review some related work on time constraint analysis.

Most of current analyses use discrete (i.e., single or multiple) activity durations and give qualitative analysis results. In [4], Li et al. extend WF-nets to Time Constraints Workflow Nets (TCWF-nets) by adding upper and lower bound constraints to transitions and places. Based on TCWF-nets, their analysis approach considers single activity duration and can only judge time constraints are satisfied or violated. In [5], Chen et al. analyze time constraints in grid workflow at run-time. They use maximum, average, and minimum activity durations to divide analysis results into four states: strong and weak consistency (satisfaction), strong and weak inconsistency (violation). In addition, in [2, 3], Eder et al. define structural and explicit time constraints and use timed activity graph to check their consistency by means of the modified Critical Path Method (CPM). In [8], they also introduce the notion of time histograms to describe a number of activity durations together with their probabilities. However, their work still presents deterministic analysis result as process’s several predicted completion time points and their probabilities. Moreover, other research qualitatively analyzes time constraints based on different modeling tools: finite timed automata [6], extend Program Evaluation and Review Technique (ePERT) [7], and workflow graph notations [9].

Some work considers probabilistic model and time constraints. Around in 1970, David Martin et al. presented the idea of probabilistically estimating graph models [20]. In [21], Egon Balas et al. discuss time schedulability problem in manufacturing environment with consideration of release dates, deadlines, and sequence-dependent setup times. Also, in [10], rather than analyze imposed time constraints, Liu et al. set the whole process and activities’ fixed-date constraints to meet their required probability of satisfaction, but they donot consider lower and upper bound constraints. In addition, our earlier works [11, 12] probabilistically analyze time constraints at workflow build-time, but they still have some deficiencies. Specifically, reference [11] proposes the idea of PTCWF-nets and attempts to analyze successful execution ratios of activities, sub-processes, and the whole process. Furthermore, reference [12] brings normal distributed activity durations and iterative workflow control structure into PTCWF-nets. Hence it presents a practicable approach to analyze single activity’s successful execution ratio. Based on this analysis result, the approach accurately adjusts time constraints to meet the activity’s required successful execution ratio by extending its deadlines. In addition, based on successful execution ratio, reference [22] introduces a novel and robust scheduling algorithm to manage actors’ personal worklist. This algorithm maintains a feasible worklist of activity instances which meet actor’s required bottom line of successful execution, and simultaneously minimizes the overall deadline violation costs in the worklist, so as to improve the reliability and performance of workflow systems. As stated above, previous probabilistic analyses concentrate on static analysis and adjustment at workflow build-time.

In conclusion, our work mainly differs from traditional research from two aspects: 1) current work either considers discrete activity durations, or only applies to single workflow control structures such as sequential and parallel control
structures. By contrast, our approach considers activities’ continuous durations and four types of workflow control structures. 2) time constraints in conventional work are either lower and upper bound constraint (e.g., in [4, 11, 12]), or fixed-date constraint (e.g., in [5] and [10]). Developing their work, our approach not only comprehensively analyzes three types of time constraints, but also reflects their dynamic changes during process execution.

VI. CONCLUSIONS

In this paper, we have proposed a dynamic approach for analyzing time constraints. The approach provides stochastic information which aims to help process managers accurately identify time constraint violations. Also, the approach dynamically updates this information at workflow run-time, which assists process managers to monitor the temporal statues of the executing business processes. To this end, the paper proposes the PTCWF-nets, which comprehensively describe three kinds of time constraints and two uncertainties in process instances, to facilitate the quantitative and dynamic analysis of time constraints. Based on PTCWF-nets, the approach first pre-analyzes all activity instances’ probabilities of satisfying time constraints when a process is instantiated. Afterwards, the approach dynamically updates these probabilities for uncompleted activity instances during process execution. Moreover, a process instance of altering contract specifications during process execution.

In our future work, we plan to extend our analysis work into multiple process instances, and investigate the relationship between run-time time constraint adjustment and cost of exception handling.

ACKNOWLEDGMENT

We thank Professor Arthur H. M. ter Hofstede’s helpful comments on paper. This work is supported by the National Basic Research Program (973 Plan) under grant No. 2009CB320700 and 2007CB310802, National High-Tech Development Program (863 Plan) under grant No.2008AA042301 and 2007AA040607, and Program for New Century Excellent Talents in University.

REFERENCES