基于时间 Petri 网的工作流模型分析*

李慧芳[1,2+], 范玉顺[1]

[1](清华大学 自动化系 CIMS 中心,北京 100084)
[2](北京理工大学 自动控制系,北京 100081)

Workflow Model Analysis Based on Time Constraint Petri Nets

LI Hui-Fang[1,2+], FAN Yu-Shun[1]

[1](CIMS Center, Department of Automation, Tsinghua University, Beijing 100084, China)
[2](Department of Automatic Control, Beijing Institute of Technology, Beijing 100081, China)
+ Corresponding author: Phn: +86-10-68912450, E-mail: huifang@bit.edu.cn, hfli@stdaily.com

Received 2002-11-25; Accepted 2003-01-28

Li HF, Fan YS. Workflow model analysis based on time constraint Petri nets. *Journal of Software*, 2004,15(1): 17~26.
http://www.jos.org.cn/1000-9825/15/17.htm

Abstract: The ultimate goal of workflow management is to implement the right person executes the right activity at the right time. To make enterprises more competitive, time-related restrictions of business processes should be considered in workflow models. A workflow model, which considers time related factors, requires time specification and verification before it goes into production so as to guarantee the time coordination in workflow executions. Through extending time attributes for the elements in WF-nets, this paper investigates the integration of the time constraints imposed on business processes into their workflow models and the new nets are called TCWF-nets. Based on analyzing the schedulability of business activities, a time consistency verification method is put forward to assure safe time interactions between activities during workflow executions. The schedulability analysis method can not only check for the time feasibility of its execution for a given workflow schedule when the time constraints are imposed on business processes, but also give an optimal schedule to guarantee the minimum duration of workflow execution for a specific case. Research results show that this method supports the time modeling and analysis in business processes, and has an important value in enhancing time management functionality as well as the adaptability to dynamic business environments of current WFMS.
Key words: workflow; time consistency; Petri nets; schedulability; verification

摘 要: 工作流管理的最终目的是实现适当的人在适当的时间执行适当的活动.企业要获得竞争力,需要在工

---

* Supported by the National Natural Science Foundation of China under Grant No.60274046 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA415340 (国家高技术研究发展计划(863))

LI Hui-Fang was born in 1965. She is an associate professor in Department of Automation Control, at Beijing Institute of Technology. Her research interests are Petri net, workflow management and scheduling optimization. **FAN Yu-Shun** was born in 1962. He is a professor and doctoral supervisor in Department of Automation, at Tsinghua University. His current research areas are enterprise modeling and workflow management, integrated platform and artificial intelligent.

© 中国科学院软件研究所 http://www.jos.org.cn

作流模型中考虑与业务过程相关的时间约束.一个考虑时间因素的工作流模型,需要在投入运行前进行时间规范与验证,以保证工作流执行的时间协调.通过为工作流网元素扩展时间属性,得到集成业务过程时间约束的工作流模型——时间约束工作流网(TCWF-nets).基于对业务活动的可调度性分析,提出了时序一致性验证方法,确保工作流执行中活动之间时间交互的安全性.在所附加的时间约束下,该可调度分析方法不仅能够检测某一给定工作流调度的时间可行性,还能对特定的实例给出一个最优调度,使工作流执行延迟最小.研究结果表明,该方法支持业务过程的时间建模与分析,对于丰富现有工作流系统的时间管理功能以及增强现存工作流软件对动态业务环境的适应性具有重要意义.

关键词: 工作流;时序一致性;Petri 网;可调度性;验证

中图法分类号: TP311 **文献标识码:** A

www.jos.org.cn

The most critical need in companies striving to become more competitive is the ability to control the flow of information and work throughout the enterprise in a timely manner. Consequently, time-related restrictions, such as bounded execution durations and absolute deadlines, are often associated with process activities and sub-processes[1]. However, arbitrary time constraints and/or unexpected delays could lead to time violations, and even cause inefficiencies or catastrophic breakdowns within business processes[2]. Therefore, dealing with time and time constraints is crucial in designing and managing business processes.

WFMSs (workflow management systems) are widely used in improving the effectivity and efficiency of business processes. However, primitive support for time management has been identified as the most significant limitations in applications of today's WFMS[3]. To satisfy the application requirements of WFMS to practical business processes, a workflow model requires not only the specification of flow but also the handling of time issues. Time management includes planning of workflow execution in time, estimating workflow duration, avoiding deadline violations and guaranteeing the satisfiability of all time constraints imposed on business processes. Proper time modeling and management will make better coordination of individual tasks and better planning of business processes possible, because early detection of time consistency of a workflow model will enable a user to predict any time-related problems, such as any violations of temporal constraints. This is particularly important for processes where any deviation from the prescribed model can be expensive, dangerous or even illegal, such as airline maintenance and hazardous material handling.

1 Background and Related Work

In the real world, all business processes exist in a temporal context and are time constrained. The comprehensive treatment of time constraints depends on time modeling of workflow processes. The existing time modeling approaches are mainly based on workflow graphs and Petri nets. Eder[4] determined timing inconsistencies at model time and found the optimal workflow execution resources at run time using the time information generated. Marjanovic[2,3] assigned a time interval to individual workflow tasks as duration constraints and checked various temporal requirements and inconsistencies of workflow systems by using the proposed verification algorithms. Ling[5] provided a time interval extension of WF-nets for the purpose of modeling and analyzing time constraint workflow systems. He put a more emphasis on checking the soundness of workflow process definitions, but considered very limited time constraints. Sadiq[6] thought the dynamism of business environment was manifested in the form of changing process requirements and time constraints, and he primarily addresses the modeling and management of changes occurring in business processes. Adam[7] has developed a PN-based approach for identifying inconsistent dependency specification in a workflow, and checking for the feasibility of its execution for a given starting time when time constraints are present. But unfortunately, they did not show how to map the time constraints imposed on business processes into the Petri net models.

These existing approaches aim to check the various time requirements and inconsistencies at build-time, and to find the optimal workflow execution resource at run-time or attempt to acquire a consistent state of workflow execution. However, they have not considered how time constraints at every workflow execution stage affect the schedulability of activities, and what is an activity decision-span when it is schedulable? On the other hand, they mainly address time representation and conflict resolution but not touch upon any performance analysis of workflow systems.

To analyze the schedulability for business activities within a time context, a state-based process modeling language (e.g. Petri nets) is required to determine when an activity is enabled and when it is executed[8]. This is very important to schedulability analysis for time constraint workflow models. In this paper, we use WF-nets[9] to model workflow systems and distinguish an activity enabling from its executing. In the subsequent sections, we introduce TCWF-nets (time constraint workflow nets), and then discuss the time modeling and time constraint satisfiability in business processes. The introduction of decision-spans for schedulable activities helps the activity agencies to manage their personal work-plans according to the global business goal. Secondly, we reveal that along a specific instance routing, the schedulability analysis is induced to a constraint-programming problem and, solving this problem gives an optimal schedule of workflow execution for this case. Finally, we also discuss the schedule-based execution.

2 Time Constraint Workflow Nets

2.1 Basic principles

The workflow model or process model is a description of the tasks, ordering, data, time, resource, and other aspects of the process. Process instance types are introduced by split nodes (decision nodes) or differenct alternative executives of a workflow. Different instances of the same process instance type are modeled by one execution subnet of workflow tasks. Obversiously, a workflow model without decision nodes represents only one process instance type, and a workflow instance is a single, individual instance of a process. A process instance is an individual enactment of a process (process instance type) with its own process data, and it is also called workflow instance when a process is represented by a workflow model. In workflow related literatures, there are no a definite discrimitation between case and instance. Case, workflow instances or process instances are all used for representing particular occurrences of the workflow or process, and activity instances are particular enactment of the activity.

Workflow nets (WF-nets) are a subclass of Petri nets that are used to model workflow processes and verify the behavioral correctness of workflows. Workflow concepts can be modeled by Petri net elements, and standard workflow building blocks, such as AND-split, AND-join, OR-split and OR-join, can be represented by various net structures. A workflow is sound if the following 3 restrictions are satisfied: 1) a workflow should always be able to complete a case; 2) every case should be completed properly, with no more work in progress after completion; and 3) every task should be executed by the workflow execution of some case. To include the notion of time, we extend a time set D and a time interval set TC for a WF-net, and a time stamp function $TOKEN_{arr}(p_i)$ for tokens in place p_i. So, a Time Constraint WF-net is

$$\text{TCWF-net}=(\text{WF-net}, TC, D)$$

where

WF-net: it is the basic workflow net system;

$TC=TC_p \vee TC_t$, where TC_p: $P \rightarrow Z \times Z$ and TC_t: $T \rightarrow Z \times Z$, Z is a set of all non-negative real numbers and $Z=\{x \in REAL | x \geq 0\}$, TC_p is a set of all place time pairs and $TC_p=\{[TC_{\min}(p), TC_{\max}(p)] \in Z \times Z | TC_{\min}(p) \leq TC_{\max}(p) \wedge p \in P\}$, TC_t is a set of all transition time pairs and $TC_t=\{[TC_{\min}(t),TC_{\max}(t)]\in Z\times Z|TC_{\min}(t)\le TC_{\max}(t)\wedge t\in T\}$. $(TC_{\min}(p),TC_{\max}(p))$ denotes the time constraints of process instances and $(TC_{\min}(t),TC_{\max}(t))$ denotes the time constraints of activity instances; D: $T\rightarrow Z$, D is a set of firing durations and $D=\{FIRE_{Dur}(t)|FIRE_{Dur}(t)\in Z\wedge t\in T\}$. $FIRE_{Dur}(t)$ denotes the executing duration of an activity instance.

In TCWF-nets, $TOKEN_{arr}(p_i)$ represents the absolute time when a process instance arrives at p_i, i.e. the global lifetime of the instance. A place time pair $(TC_{\min}(p_i),TC_{\max}(p_i))$ denotes the time period during which the subsequent transition can be enabled when a case arrives at p_i, that is, the time interval during which the instance is staying at p_i. A transition time pair $(TC_{\min}(t_j),TC_{\max}(t_j))$ denotes the firable time interval during which the activity corresponding to t_j can be executed after t_j is enabled. Thus, the execution time of an activity depends not only on $TOKEN_{arr}(p_i)$, but also on $TC_{\min}(p_i)/TC_{\max}(p_i)$ and $TC_{\min}(t_j)/TC_{\max}(t_j)$, where p_i belongs to the set of input places of t_j. $FIRE_{Dur}(t_j)$ represents how long the execution of activity t_j lasts.

Comparing with the existing time-related Petri nets[5,10], TCWF-nets include more general time constraints, such as place time pairs, transition time pairs, firing durations of transitions, and token arrival times, which describe the enabled interval of activities, the executable interval of activities, the execution duration of activities and the arrival time of a specific case. The weakly firing mode[10] in TCWF-nets suits for modeling an activity, enabling and executing in workflow systems. Whether an enabled transition can be fired depends on the additional firing mechanisms. For an automatic activity, it can be executed immediately once it is enabled, but for a non-automatic activity, it can be triggered by a human participant, etc. Due to considering the execution duration of an activity, all the tokens used for enabling the activity are preserved during its execution. Once it completes execution, it generates tokens to its output places with the corresponding number and thus, the workflow proceeds. For facilitating analysis, we use $FIRE_{enable}(t_j)$ for denoting the time when t_j is enabled, $EEBT(t_j)/LEET(t_j)$ for denoting the earliest/latest enabling time, $EFBT(t_j)/LFET(t_j)$ for denoting the earliest fire beginning/latest fire ending time, $FIRE_{begin}(t_j)/FIRE_{end}(t_j)$ for denoting the actual fire beginning/ending time, $I_t(p_i)/O_t(p_i)$ for denoting the set of input/output transitions of p_i, and $I_p(t_j)/O_p(t_j)$ for denoting the set of input/output places of t_j.

2.2 Time constraint modeling

In order to represent time information, we need to augment the workflow model with the following basic temporal types: time point, duration and interval constraints. Given a workflow schema, a workflow designer can assign execution durations, $FIRE_{Dur}(t_j)$ for individual activities or the whole workflow process, as well as relative enabled intervals $(TC_{\min}(p_i),TC_{\max}(p_i))$ for places, relative executable intervals $(TC_{\min}(t_j),TC_{\max}(t_j))$ for activities, and even arrival times $TOKEN_{arr}(p_i)$ for a specific case. These durations and intervals can be either calculated from past executions, assigned by specialists based on their experience and expectation or assigned according to the resource loads or the urgency of activities. Activity executable intervals correspond to the allowable execution times of activities after they are enabled by a process instance. At build-time, these intervals are specified relative to the arrival time of process instances, and at process instantiation time, these relative intervals can be converted to absolute time intervals, and the time consistency of coorsponding process instances can be statically verified. In the sequel, the consistency of these time constraints is monitored at run-time.

• The arrival time of a process instance, $TOKEN_{arr}(p_j)$ is an absolute time representing the global lifetime of a case arriving at p_j. For the start place i, it is the startup time of the process; and for the others, it is the time when the immediate preceding activity of p_j completes firing;

• The interval $(TC_{\min}(p_i),TC_{\max}(p_i))$ denotes the time period during which p_i's succeeding activities are enabled after a case arrives at p_i. It also represents how long an instance will stay in place p_i and it may be fallen into $(0,\infty)$ under different cases. This interval includes the waiting time and servicing time of a case;

• The interval ($TC_{\min}(t_j)$,$TC_{\max}(t_j)$) denotes the time period during which the activity corresponding to t_j can be executed after it is enabled. If t_j is enabled at T_0, then it can be executed within the absolute interval ($T_0+TC_{\min}(t_j)$, $T_0+TC_{\max}(t_j)$). So ($T_0+TC_{\min}(t_j)$,$T_0+TC_{\max}(t_j)$) is called an absolute execution interval of t_j, where T_0 denotes when t_j is enabled;

• The firing duration $FIRE_{Dur}(t_j)$ which corresponds the execution duration of an activity, denotes how long the firing of t_j will last.

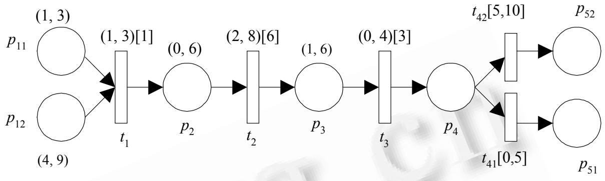

Fig.1 Fragment of a simple TCWF-net

For instance in Fig.1, $TOKEN_{arr}(p_{11})$ denotes when a case arrives at p_{11}. The interval (1,3) associated with p_{11} denotes a time range during which the case will enable transition t_1 after it arrives at p_{11}. If a case arrives in p_{11} at $TOKEN_{arr}(p_{11})$, then t_1 will be enabled during the absolute time range ($TOKEN_{arr}(p_{11})+1$,$TOKEN_{arr}(p_{11})+3$). The (2,8)[2] means that t_2 can be executed during the interval (2,8) after it is enabled by p_2 and, $FIRE_{Dur}(t_2)=6$ means that the firing duration of t_2 is 6 time units.

It is important to note the fact that if a task can be executed for a specific case, then this does not mean that the task is executed directly. For example, if a task is to be executed by an employee, then the employee has to be available and willing to execute the task. If the employee is ill or on holiday, then the task will not be executed. The workflow management is not in a complete control, but just supports the workflow. Since the enabling of a task does not imply that the task will be immediately executed, it is crucial to discriminate between the activity enabled interval and execution interval. Therefore, a triggering mechanism is required for leading to the execution of an enabled task. TCWF-net uses a weak firing mode that means an enabled activity can be fired at any time during its execution interval.

This paper exactly describes the activity enabled interval and execution interval. This distinction is beneficial because we aim at determining whether an activity can successfully complete firing within a finite execution interval and identifying time violations when the activity fails to complete firing. Consequently, we will provide some useful suggestions for system designers to design time safe and reliable WFMS.

3 Time Constraint Satisfiability

After activity durations $FIRE_{Dur}(t_j)$, case arriving times $TOKEN_{arr}(p_i)$, activity enabled intervals ($TC_{\min}(p_i)$, $TC_{\max}(p_i)$) and the relative execution interval ($TC_{\min}(t_j)$,$TC_{\max}(t_j)$) are designed, time calculations are required for converting the relative time intervals to absolute time points, computing the earliest enabled beginning time $EEBT(t_j)$ and latest enabled ending time $LEET(t_j)$, computing the earliest fire beginning time $EFBT(t_j)$ and latest fire ending time $LFET(t_j)$, and so on. When $EEBT(t_j)$/$LEET(t_j)$ and $EFBT(t_j)$/$LFET(t_j)$ are obtained, the workflow designer can use the following definitions and theorems to assert whether an activity t_j can complete its firing within a finite time interval. Also, the designer can re-assign some time constraints at process instancing time for a specific case, and then re-analyze the schedulability for activities or the whole process.

3.1 Time related calculations

The purpose of time modeling lies in managing time in workflows[4], i.e. determining workflow execution planning in time to guarantee the time safety in execution. The precondition of the safe execution of workflows is the satisfiability of all time constraints imposed on the process. If all time constraints satisfy a process definition, then they are consistent with the workflow process logic or the workflow is global consistent. To avoid the possible

www.jos.org.cn

conflict between time constraints, it is required to find a workflow execution that satisfies time constraints imposed on the business process. So the case startup time must be considered. In a TCWF-net model, the enabled intervals and execution intervals of activities are restricted. Selecting different firing times may cause that an activity fails to complete the firing within a restricted enabled time interval. That is, there exists an activity which may fail to successfully complete an execution. TCWF-net and WF-net have different reachability, and a state that is reachable in a WF-net modeling is not necessarily reachable in TCWF-net modeling because of the imposed time constraints.

Definition 1. An enabled activity t_j is said to be firable in a marking M in a TCWF-net, if $LFET(t_j)\geq EFBT(t_j)$; t_j is said to be able to complete firing successfully, if $LFET(t_j)-EFBT(t_j)\geq FIRE_{Dur}(t_j)$.

In a marking M, if an activity can successfully complete firing, it is *schedulable*, i.e. the activity execution satisfies the process time constraints. The schedulability analysis for time constraint workflow models is to assert the time feasibility in execution for a specific case. Therefore, all analysis is based on the arriving of a case. An activity instance can be fired only when its corresponding process instance arrives, and the workflow process is progressed only after the activity finishes. To analyze the schedulability for an activity t_j, firstly it needs to compute the time pair $EEBT(t_j)/LEET(t_j)$, considering the global time of workflow instances. We have the following formulae[11], where $p_i\in I_p(t_j)(i=1,2,...,k)$:

$$EEBT(t_j)=\underset{i}{\mathrm{MAX}}\,[TOKEN_{arr}(p_i)+TC_{\min}(p_i)] \tag{1}$$

$$LEET(t_j)=\underset{i}{\mathrm{MIN}}\,[TOKEN_{arr}(p_i)+TC_{\max}(p_i)] \tag{2}$$

Theorem 1. In a TCWF-net marking M, for an enabled activity (transition) t_j, $EFBT(t_j)/LFET(t_j)$ can be calculated by

$$EFBT(t_j)=EEBT(t_j)+TC_{\min}(t_j) \tag{3}$$

$$LFET(t_j)=LEET(t_j) \tag{4}$$

Proof. A transition t_j is firable, if and only if it is enabled by each of its input places, and it can not be fired until $TC_{\min}(t_j)$ time interval elapsing after it is enabled. If t_j is enabled at time T_0, then the firable time of t_j which is denoted as τ, should satisfy (5) as follows:

$$T_0+TC_{\min}(t_j)\leq\tau\leq T_0+TC_{\max}(t_j) \tag{5}$$

Since T_0 denotes the enabled time of t_j, it should satisfy (6) as follows:

$$EEBT(t_j)\leq T_0\leq LEET(t_j) \tag{6}$$

By replacing T_0 with (6) in (5), we can easily have (7) as follows:

$$EEBT(t_j)+TC_{\min}(t_j)\leq\tau\leq LEET(t_j)+TC_{\max}(t_j) \tag{7}$$

The t_j will stop firing immediately as soon as any of its input places stop enabling it, so

$$\tau\leq LEET(t_j) \tag{8}$$

Combining (7) and (8), we can get $\tau_{\min}=EEBT(t_j)+TC_{\min}(t_j)$, $\tau_{\max}=LEET(t_j)$, where $\tau_{\min}$ and $\tau_{\max}$ are the earliest fire beginning time $EFBT(t_j)$ and the latest fire ending time $LFET(t_j)$ of t_j, respectively. So, we conclude: $EFBT(t_j)=EEBT(t_j)+TC_{\min}(t_j)$ and $LFET(t_j)=LEET(t_j)$.

In fact, $TC_{\min}(t_j)/\mathrm{TC}_{\max}(t_j)$ is only a relative time pair, if and only if an enabled time τ of t_j during the interval $(EEBT(t_j),LEET(t_j))$ has been determined, and it constrains the firable beginning and ending time of t_j. That is, t_j must start its firing at time $\tau+TC_{\min}(t_j)$ then end its firing at $\tau+TC_{\max}(t_j)$. The different selecting of τ may lead to two different cases: $\tau+TC_{\max}(t_j)\leq LEET(t_j)$ and $\tau+TC_{\max}(t_j)\geq LEET(t_j)$. The former implies that t_j can complete its firing successfully, but the later implies that t_j will be disabled by its input places before $\tau+TC_{\max}(t_j)$. So, the latest fire ending time $LFET(t_j)$ is independent of $TC_{\max}(t_j)$ but depends only on t_j's latest enabled ending time $LEET(t_j)$.

© 中国科学院软件研究所 http://www.jos.org.cn

It is required to be pointed out that, for supporting global distributed business applications, time-dependent factors have to be incorporated into both the build-time process definition and the run-time management system. The reason is that activities of the workflow may belong to different time zones, flow will also take certain tranmission durations from one site to another, e.g., the duration of the material flow for delivering products from the supplier to the customer. In order to maintain global time of a process instance, the flow duration and time difference conversion must be considered in the workflow model when computing the arrival time of a process instance, and this is needed to be analyzed for different workflow structures, such as OR-split, OR-join, AND-split and AND-join.

3.2 Time consistency verification

In order to guarantee the correct execution of time constraint workflow instances, it is required to check the time constraint satisfiability for a process definition or to find the possible conflicting between the time constraints and workflow execution routings. The schedulability of t_j can be asserted by using Eqs.(1)~(4) and definition 1. The time consistency for activity execution depends directly on its schedulability and, a schedulable activity means that the process time constraints satisfy the activity execution logic. So, the key of time consistency verification for a TCWF-net model lies in the schedulability analysis for the process model. There are 2 cases in time consistency verification:

• For an individual activity t_j, if time pair $EFBT(t_j)$/LFET(t_j) calculated by Eqs.(1)~(4) satisfies $LFET(t_j)-EFBT(t_j)\geq FIRE_{Dur}(t_j)$, then t_j is schedulable, i.e. t_j can complete its firing under the imposed time constraints;

• For a workflow instance, the precondition that the execution of this instance satisfies all imposed time constraints is that all activities belonging to the instance are schedulable with respect to the initial marking of the case. So we have definition 2 as the following:

Definition 2. A workflow system (or a instance logic of the process) is time consisten*t*, if and only if all activity instances along this instance logic are schedulable with respect to the initial state of the workflow.

For a process instance denoted by an activity sequence $\sigma=M_0t_1M_1t_2M_2...t_iM_i...t_nM_n$ or $t_1t_2...t_i...t_n$, verifying its schedulability needs to assert whether all activities belonging to σ are schedulable by using Eqs.(1)~(4), because the workflow system denoted by σ is schedulable or time consistent if and only if all activities belonging to σ are schedulable. Firstly, it is required to compute the time pair $EFBT(t_j)/LFET(t_j)$ for j=1,2,...,n in turn, then assert the schedulability of t_j by Definition 1. If t_j is schedulable, then fire t_j and compute $EFBT(t_{j+1})/LFET(t_{j+1})$ according to the fire ending time of t_j, $FIRE_{end}(t_j)$. Only if each t_j in σ is schedulable, this process instance is time consistent.

3.3 Handling of time violations

If an activity t_j is nonschedulable, our method also can identify the cause of time violations. Usually there are 2 methods for treating time violations:

• For a nonschedulable activity t_j, firstly we relax the time constraints of places or transitions occurring prior to t_j, then we re-compute the time pair $EFBT(t_j)/LFET(t_j)$ and assert the schedulability of t_j according to Definition 1. This may be done several times till t_j is schedulable;

• Certainly, for those important time constraints that are not allowed to modify or cases where time constraint relaxation still cannot change the schedulability of t_j at all events, it is required to analyze workflow semantics and then modify the structure of TCWF-nets in order to re-obtain the schedulability of t_j. Subsequently, analyzing the schedulability of t_j under the new workflow model may be repeated till t_j becomes schedulable. Of course, the above 2 methods can also be used alternately to make t_j schedulable.

It needs to be pointed out that this is a static designing method, which is used to locate time violations and then provide suggestions on modifying the system specifications and/or relaxing the time constraints for the system

© 中国科学院软件研究所 http://www.jos.org.cn

designers.

4 Schedulability Analysis Method

For a workflow instance with a given planning and a set of time constraints, our time consistency verification method aims to simulate the time reachability in execution and analyze the schedulability for each activity, then verify the time consistency between the time constraints and process model. If all activities along σ can complete their firing within limited time intervals, then these time constraints are reasonable or the workflow instance is time feasible in execution.

4.1 Decision-Span for a schedulable activity

To analyze the schedulability of an activity t_j, we need to know the absolute enable time of t_j. For instance in Fig.1, we can determine when t_2 is enabled only if t_1 ends its firing. Moreover, $TOKEN_{\rm arr}(p_2)$=$FIRE_{end}(t_1)$= $FIRE_{begin}(t_1)$+$FIRE_{Dur}(t_1)$. To determine $FIRE_{begin}(t_1)$, a new notation decision-span is introduced to denote the firable decision-span of a schedulable activity:

Definition 3. In a marking M in a TCWF-net model, for a schedulable activity t_j, if $D(t_j)$ denotes the decision-span of t_j and $UD(t_j)$ denotes the upper bound of $D(t_j)$, then the schedulable decision-span of t_j can be expressed as $0 \le D(t_j) \le UD(t_j)$, where

$$\mathrm{UD}(t_j)=\mathrm{LFET}(t_j)-\mathrm{EFBT}(t_j)-\mathrm{FIRE}_{\mathrm{Dur}}(t_j) \tag{9}$$

So the actual firing time of t_j is: $FIRE_{begin}(t_j)=D(t_j)+EFBT(t_j)$. It makes sure that t_j can successfully complete execution as long as t_j is fired within the allowable range of $D(t_j)$. For $\forall p_i \in O_p(t_j)$, there exists $TOKEN_{arr}(p_i)=FIRE_{end}(t_j)$. It is easy to see that the time when the instance arrives in p_i depends directly upon the selection of $D(t_j)$, and at the same time on the multi-factors such as resource loads, activity emergency and the influence on its succeeding activities, etc.

4.2 Schedulability analysis along a specific path

For instance in Fig.1, $D(t_1)\in(0,UD(t_1))=(0,LFET(t_1)-EFBT(t_1)-FIRE_{Dur}(t_1))$ and $UD(t_1)$ can be determined by the time constraints of p_{11}, p_{12} and t_1. Once t_1 completes its firing, we have $FIRE_{end}(t_1)=TOKEN_{arr}(p_2)=f_1(D(t_1))$, and more $EFBT(t_2)/LFET(t_2)=g_1(D(t_1))/g_2(D(t_1)),FIRE_{end}(t_2)=TOKEN_{arr}(p_3)=D(t_2)+EFBT(t_2)+FIRE_{Dur}(t_2)=f_2(D(t_1),\mathrm{D}(t_2))$. In the same way we can get $FIRE_{end}(t_3)=TOKEN_{arr}(p_4)=f_3(D(t_1),D(t_2),D(t_3))$ in turn, where some of f_1, f_2 and f_3,..., may be nonlinear functions depending on different TCWF-net structures. This schedulability analysis is a piecewise decision-making process, and the selecting activity decision-spans may be nonlinear for general TCWF-net topology structures.

A workflow instance corresponds to a reachable path σ in TCWF-net models, each decision-span constraint of transitions in σ can be determined by Eqs.(1)~(4) and (9), i.e. $0 \le D(t_j) \le UD(t_j)$. There may exist nonlinear constraint relations between $D(t_1),D(t_2),...,D(t_n)$ for general TCWF-net structures, and these constraint relations are denoted as:

$$f_k(CC,DT)\ge 0,\ k=1,2,\ldots,n \tag{10}$$

where CC represents the set of time constraints including $TC_{\min}(pt_k)/TC_{\max}(pt_k)$ and $FIRE_{Dur}(t_j)$, $DT=(D(t_1),D(t_2),\ldots, D(t_n))$ denotes the decision-span vector.

Besides guaranteeing time feasible in workflow execution, it is required to calculate the *optimal* DT^* to achieve the optimal workflow execution. Here, this optimization problem is:

$$D=\{DT, f(j,DT)\ge 0,\ j=0,1,\ldots,n-1\} \tag{11}$$

For minimizing $J(DT)=FIRE_{end}(t_n)$, where t_n is the last activity of its corresponding process instance and $FIRE_{end}(t_n)$ denotes the earliest finishing time of the instance, the scheduling problem becomes into solving the following programming:

$$\begin{cases} \underset{DT}{\text{Min}}\, J(DT) \\ \text{s.t.} \quad f(j,DT) \geq 0, \quad j=0,1,\ldots,n-1 \end{cases} \tag{12}$$

The decision-span vector DT can be determined by finding the solution to the problem formulated as (12), which is subjected to the constraint (10). For an actual application context, using different methods can solve the above programming.

It needs to point out that in a TCWF-net-based workflow analysis, the selection between normal operation and time-violation-related exception handling can be denoted by assigning different firing deadlines for OR-split activities[11], and an activity with earlier deadline has higher priority for firing. Here, schedulability analysis must consider firable priorities. For example in Fig.1, t_{41} is enabled earlier than t_{42} when a case arrives at p_4, i.e. t_{41} has a higher priority for firing. If t_{41} can not finish, the token used for enabling t_{41} will be put back to p_4 and its arrival time will be modified as: $\tau = FIRE_{end}(t_{41f})+1=f(\mathrm{D}(t_{41f}))$. This token can be still used to enable t_{41}. The precondition that t_{42} can be fired is that t_{41} fails to complete its firing within a specified time limit, so the instance routing from $M(p_2)$ to $M(p_{52})$ is $\sigma_{52}=t_2t_3t_{41f}t_{42}$, where t_{41f} means that t_{41} fails to finish within its limited time interval. Once the constraint conditions of every activity decision-span along σ_{52} are determined, the schedulability analysis can also be formulated as the above programming shown in (11)~(12).

4.3 Schedule-Based execution

The execution of a workflow instance can start once all activities belonging to this instance are verified to be schedulable under imposed time constraints. All $EFBT(t_j)/LFET(t_j)$ (j=1,2,...,n) specifies an absolute range for activity execution times such that there *exists* a combination of activity beginning and finishing times that satisfies all time constraints and where each beginning and finishing time is within the range $[EFBT(t_j),LFET(t_j)]$ of its corresponding activity.

We define a schedule to be an activity sequence $S=\{t_1/D(t_1)/EFBT(t_1)/LFET(t_1),t_2/D(t_2)/EFBT(t_2)/LFET(t_2),\ldots, t_i/D(t_i)/EFBT(t_i)/LFET(t_i),\ldots,t_n/D(t_n)/EFBT(t_n)/LFET(t_n)\}$, which consists of activities and their execution-related time attributes. This sequence gives the decision-span $D(t_j)$ and the earliest fire beginning/latest fire ending time $EFBT(t_j)/LFET(t_j)$. Thus, any combination of actual execution beginning/ending times of all activities within $[FIRE_{begin}(t_j),FIRE_{end}(t_j)]$ ranges satisfies all time constraints imposed on the business process, where $FIRE_{begin}(t_j)$ and $FIRE_{end}(t_j)$ are $FIRE_{begin}(t_j)=D(t_j)+EFBT(t_j)$ and $FIRE_{end}(t_j)=FIRE_{begin}(t_j)+FIRE_{Dur}(t_j)$. In other words, given a schedule or a process instance, no violations of time constraints occur as long as each activity t_j finishes its execution at time within the interval $[FIRE_{begin}(t_j),FIRE_{end}(t_j)]$. So, the build-time process definition of a TCWF-net consists of the following steps:

• Design the workflow process at the logical level, i.e., define the traditional workflow process;

• Determine the time constraints of the modeled business process;

• Map the above time constraints to the time attributes of the elements in a TCWF-net, such as $TC_{\min}(p_i)/TC_{\max}(p_i),TOKEN_{arr}(p_i),TC_{\min}(t_j)/TC_{\max}(t_j)$ and $FIRE_{Dur}(t_j)$;

• Check the reasonability of these time constraints or verify the time consistency with the process definition;

• If there exist some time constraint violations, then relaxing time constraints and modifying the workflow structure are required in order to make sure all activities are schedulable. Finally we know all $UD(t_j)/EFBT(t_j)/LFET(t_j)$;

• Along the workflow instance, find all constraints between all activity decision-span $D(t_j)$ according to different TCWF-net structures, and denote the optimization problem as (11)~(12) shown in Section 4.2;

• Solve the above programming to get the optimal schedule $S=\{t_1/D(t_1)/EFBT(t_1)/LFET(t_1),t_2/D(t_2)/EFBT(t_2)/$

© 中国科学院软件研究所 http://www.jos.org.cn

$LFET(t_2),...,t_i/D(t_i)/EFBT(t_i)/LFET(t_i),...,t_n/D(t_n)/EFBT(t_n)/LFET(t_n)\}$. Carrying the instance into execution as this schedule will lead to the minimum duration of the whole process.

In a word, our method not only guarantees that time constraints are properly modeled and it provides a schedule *S* of a workflow instance to make sure that all activities are time feasible in execution, but also designs an optimal and schedulable TCWF-net model for a specific instance through a circular process "modeling→ verifying→modifying→re-verifying→re-modifying...".

5 Comparison and Conclusions

Starting from the requirements of time management in enterprise workflows and basing on analyzing the limitation of existing time management methods, a new time modeling and analysis method is put forward in this paper to maintain the safe execution of time constraint workflow instances. Comparing with existing time modeling methods, the major advantages of our method consist of four aspects. Firstly, the enabled intervals and execution intervals of activities are distinguished and taken into account in the proposed approach, while it is neglected in the other approaches. This consideration is significant because it enables the proposed approach to analyze the effect of temporal constraints at every stage of workflow execution on the schedulability of activities. Secondly, based on schedulablity analysis for activities and appropriate handling of time violation, time constraints can be validly modeled and the time consistency in workflow execution can be verified. But this is not referred to in existing approaches. Thirdly, the decision-span $D(t_j)$ represents the urgency of activity t_j to some extent, and this facilitates the workflow participants to manage their work plans according to the whole business goal. Fourthly, solving the given non-linear programming can find the optimal time planning for workflow execution denoted as $D(t_1)/EFBT(t_1)/LFET(t_1),D(t_2)/EFBT(t_2)/LFET(t_2),...,D(t_j)/EFBT(t_j)/LFET(t_j),...,D(t_n)/EFBT(t_n)/LFET(t_n)$. This enables the time analysis approach can be used for analyzing the time performance of a workflow, because it can result in a somewhat optimal schedule guaranteeing a minimal duration for the execution of a workflow execution in specific cases. Research results demonstrate that TCWF-net-based time modeling and analysis not only supports the time management in business processes and implements time coordination between activities, but also is crucial to promote the application of workflows to complex enterprises.

References:

[1] Li HF, Fan YS. Overview on managing time in workflow systems. Journal of Software, 2002,13(8):1552~1558 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1552.pdf

[2] Marjanovic O, Orlowska ME. On modeling and verification of temporal constraints in production workflows. Knowledge and Information Systems, 1999,1(2):157~192.

[3] Marjanovic O. Dynamic verification of temporal constraints in production workflows. In: Proc. of the 11th Australasian Database Conf. (ADC2000). Canberra, 2000. 74~81.

[4] Eder J, Panagos E, Pozewaunig H, Rabinovich M. Time management in workflow systems. In: Abramowicz W, Orlowska ME, eds. Proc. of the 3rd Int'l. Conf. on Business Information Systems. Heidelberg, London, Berlin: Springer-Verlag, 1999. 265~280.

[5] Ling S, Schmidt H. Time Petri nets for workflow modeling and analysis. In: Proc. of IEEE Int'l Conf. on System, Man and Cybernetics. 2000. 3039~3044.

[6] Sadiq SW, Marjanovic O, Orlowska M. Managing change and time in dynamic workflow processes. Int'l. Journal of Cooperative Information Systems, 2000,9(1&2): 93~116.

[7] Adam NR, Atluri V, Huang WK. Modeling and analysis of workflow using Petri nets. Journal of Intelligent Information Systems, 1998,10(2):131~158.

[8] De Michelis G. Net theory and workflow models. In: Donatelli S, Kleijn J, eds. Proc. of the 20th International Conf. on Application and Theory of Petri Nets. LNCS 1639, Berlin, New York: Springer Verlag, 1999. 282~283.

[9] Van der Aalst WMP. The application of Petri nets in workflow management. The Journal of Circuits, Systems and Computers, 1998, 8(1): 21~66.

[10] Tsai JJP, Yang SJ. Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications. IEEE Trans. on Software Engineering, 1995,21(1):32~49.

[11] Li HF, Fan YS. Schedulability analysis method of timing constraint Petri nets. Tsinghua Science and Technology, 2002,7(6): 596~601.

附中文参考文献:

[1] 李慧芳,范玉顺.工作流系统时间管理.软件学报,2002,13(8):1552~1558.http://www.jos.org.cn/1000-9825/13/1552.pdf

www.jos.org.cn