Time Petri Nets for Workflow Modelling and Analysis

Sea Ling and Heinz Schmidt
School of Computer Science & Software Engineering
Monash University
P.O. Box 197, Victoria 3145, Australia

Abstract
Time management in workflow processes is crucial in determining and controlling the life cycle of business activities. In our model, a temporal interval as an execution duration is assigned to every workflow task. While the real time taken by the task is non-deterministic and unpredictable, it may be between the bounds, thus specified. We extend Workflow nets (WF-nets) with time intervals and call the new nets Time WF-nets (TWF-nets). Extending our previous results on timed Petri nets, we show that certain behavioural properties of workflow processes modelled in TWF-nets can be verified. Using a clinical health care process as a case study, we also illustrate the modelling of shared resources available at different times.

1 Introduction
The main purpose of workflow management systems [1] is to support the definition, execution and control of workflow processes. A workflow process defines a set of activities and the specific order they are to be executed to achieve a common goal. The co-ordinated (sequential and/or parallel) set of activities are viewed as a workflow graph.

Time management in workflow processes is crucial in determining and controlling the life cycle of business activities. Most integrations of time management with workflow involve the assignment of deadlines and other external temporal and synchronisation constraints, the calculation of the overall process duration and the checking of timing inconsistencies [2, 3]. None has dealt with the effect of temporal constraints at every stage of workflow execution and how these constraints affect the liveness and safety aspects of the entire process. To perform these studies, one requires a workflow representation language that is state-based rather than event-based. Most existing workflow management systems are however event-based.

The semantics of workflow graphs has been modelled by the Petri net language [4], which is widely used as a process specification and verification language [5, 6]. A good reason for using Petri nets to model workflows is that Petri nets treat states and events on an equal footing [7]. They are event-based and state-based at the same time. Their formal analysis methods allow extraction and calculations on either of these aspects in isolation or in combination where appropriate. This results in a class of Petri nets called Workflow nets (WF-nets) [6]. States are represented explicitly and the model is able to differentiate between the enabling and the execution of a task. The distinction is crucial for our study with time because we want to determine when the task is enabled, when it is executed and for how long the execution will last.

In our model, we assign a temporal interval as a duration constraint to every workflow task. While the duration taken by the task is non-deterministic or unpredictable, it may fall between the bounds, thus specified. An interval time extension of WF-nets is then used to specify the workflow with such time constraints.

Although the safety property in Petri nets is closely related to the boundedness property, in our work we define safety in relation to a phenomenon called contact-freeness (free of contact situations) [8] in Elementary Net (EN) systems [9]. This notion of safety is incorporated with time and applied to the time extension of WF-nets. The notion is called time safety, which allows us to model and control multiple threads of process execution competing for available but limited resources provided for the performance of a given task.

Using a workflow example, we show that the existing verification of behavioural properties, such as liveness, and soundness in untimed WF-nets, can be readily applied to workflow systems with time constraints.

In this paper, after introducing some Petri net terminology, we describe Time Workflow nets and the associated notions of safety, liveness and soundness. We
then model a clinical health care process [10], whose tasks are each assigned a time interval, modelling a range of possible durations.

2 Petri Net Preliminaries

This section introduces some basic notions and notations of Petri nets to be used in later sections. They are based on the standard class of Petri nets called Place/Transition (P/T) nets. For a more detailed description of P/T nets, readers should refer to existing work in the literature [11].

Given a set of identifiers U, a net structure (or net) N is a tuple (P, T, F), where P \subseteq U and T \subseteq U are non-empty, finite disjoint sets of places (S-elements) and transitions (T-elements) respectively, and F \subseteq (P \times T) \cup (T \times P), is the flow relation. NE = P \cup T is called the set of net elements. The components of a net N are also denoted by PN, TN and FN.

A triple (P', T', F') is a subnet of N if P' \subseteq P, T' \subseteq T and F' = F \cap((P' \times T') \cup(T' \times P')). If X is a subset of net elements of N, then the triple (P \cap X, T \cap X, F \cap (X \times X)) is called a subnet of N generated by X.

For some x \in NE, the set x* = \{ y \mid (y,x) \in F \} is the preset of x and the set x* = \{ y \mid (x,y) \in F \} is the postset of x. For a set X \subseteq NE, x* = \bigcup x \in X x* and X* = \bigcup x \in X x*.

A path of a net (P, T, F) is a nonempty sequence z_1, z_2, ..., z_k (k \in \text{Nat}) of net elements which satisfies (z_1, z_2), ..., (z_{k-1}, z_k) \in F. The net is weakly connected (or just connected) if every two nodes x, y satisfy (x, y) \in (F \cup F^{-1}). It is strongly connected if (x, y) \in F^*, where F^* is the reflexive and transitive closure of F, i.e., for every two net elements x, y there is a path leading from x to y.

A marking of a net structure (P, T, F) is a mapping M : P \rightarrow \mathbb{N}. It is represented by a vector (M(p_1), M(p_2), ..., M(p_n)), where p_1, p_2, ..., p_n is an arbitrary fixed enumeration of P. A Petri net system is a pair (N, M_0) where N is a net structure and M_0 is a marking called the initial marking of the system.

A net N is an S-net if \text{\vert x* \text{\vert = \text{\vert x* \text{\vert = 1 for every transition t. It is free-choice if (p, t) \in F_N implies x* \subseteq F_N for every place p and every transition t. A system (N, M_0) is free-choice if N is free-choice.}

N is asymmetric-choice if for every two places p_1 and p_2, either p_1* \cap p_2* = \emptyset or p_1* \subseteq p_2* or p_2* \subseteq p_1*. (N, M_0) is asymmetric-choice if N is asymmetric-choice.

Let N' be the subnet of N generated by a nonempty set A \subseteq NE of net elements. N' is an S-component of N if p \cup p* \subseteq A for every place p of A, and N' is an S-net. The S-component is strongly connected if N' is strongly connected.

Recall also that a set R of places is a siphon if \text{\bullet R \subseteq \bullet R}. It is a trap if \text{\bullet R \subseteq \bullet R}. A siphon or a trap is called proper if it is not the empty set.

3 Time Workflow Nets

In our previous work, a class of Petri nets, called Elementary Net (EN) systems [9], was enriched with timing information. Among the different classes of Petri nets, including P/T nets, the EN model is chosen as the underlying net model for the time extension of WF-net because EN systems can contain at most one token in every place and are trivially 1-bounded. Furthermore, our notion of safety is derived from the concept of contact situation which exists in this type of net systems.

The time extension of EN systems called Time Elementary Net (TEN) systems serves as the formalism, from which the definition of Time Workflow nets is formulated. Some of these notions are recalled from our previous work [12, 13], in particular the notions of liveness, safety and time safety.

We recall workflow nets, as defined in the literature [6].

A net N = (P, T, F) is a WF-net (WorkFlow net) if and only if (i \in P \land i = \emptyset) \land (o \in P \land o^* = \emptyset) and the short-circuited net (P, T \cup \{t\}, F \cup \{(o, t), (t, i)\}) is strongly connected, where t \in UT. Place i is the input place and place o is the output place of the WF-net.

To include the notion of time, we define a time set and a time interval, in a similar fashion as [14]. A time set is a set of all non-negative reals: TS = \{ x \in \text{REAL} \mid x \geq 0 \}. The set of time intervals is defined as INT = \{ [y, z] \in TS \times TS \mid y \leq z \}. If x \in TS and [y, z] \in INT, then x \in [y, z] iff y \leq x \leq z. Also, x \in [y, z] iff x = y. The interval precedence relation \subseteq is defined as -\subseteq - INT \times INT such that for each [a_1, b_1], [a_2, b_2] \in INT, [a_1, b_1] \subseteq [a_2, b_2] iff b_1 < a_2.

The WF-net definition is extended with time intervals for a Time Workflow net (TWF-net) to suit the real-time context. A Time WF-net (TWF-net) N is a tuple (P, T, F, FI) such that: (P, T, F) is a WF-net and FI associates each transition t \in T with a static firing interval: FI : T \rightarrow INT. The interval, FI(t) = [min_t, max_t], is a pair of real numbers
referred to as the minimum static firing time and the maximum static firing time respectively. A TWF system is a tuple \((N,M_0)\) where \(N\) is a TWF-net and \(M_0\) is an initial marking.

A state of a TWF system is defined as a marking in time. It is a pair \((M,\overline{FL})\), where \(M \subseteq P_N\) is a marking; and \(\overline{FL}\) associates each transition \(t\) with a dynamic firing interval: \(\overline{FL} : T_N \rightarrow INT\). The interval, \(\overline{FL}(t) = [\overline{min}_t, \overline{max}_t]\), is a pair of real numbers referred to as the earliest dynamic firing time and the latest dynamic firing time respectively. The initial state is denoted by \(S_0 = (M_0, F_0)\) with initial marking \(M_0\) and \(F_0(t) = \{\min_t, \max_t\}\) for each enabled transition \(t\). The dynamic firing interval defines a pair of time values between which \(t\) is allowed to fire.

A transition \(t \in T_N\) is enabled in a state \(s = (M, \overline{FL})\) if \(*s \subseteq M\). \(t\) is firable if \(t\) is enabled and the following condition holds: \(*s \cap M = \emptyset\). Note that the last condition ensures that \(t\) does not fall into a contact situation. The firing time of a firable transition \(t\) is denoted by \(\tau(t)\), i.e., \(\tau(t) \in \overline{FL}(t)\). \(t\) will fire at state \(s\) after a legal firing time, \(\tau(t)\), defined as \(\overline{min}_t \leq \tau(t) \leq \min\{\overline{max}_t, \tau(t)\} \mid t' \neq \tau \land t' \in s\).

When \(t\) fires in a given state \(s_1\) after a legal firing time, \(\tau(t)\), a new state \(s_2\) is produced through the following changes: (1) a change in the marking: a token is removed from every \(p \in *t\); and a token is added to every \(p \in *t\). (2) a change in the dynamic firing intervals: For each transition \(t'\) enabled in \(s_2\), the dynamic firing interval is shifted left by \(\tau(t)\): \(\overline{min}_t = \max\{\overline{min}_t, \tau(t) - \tau(t)\}\), and \(\overline{max}_t = \max\{\overline{max}_t, \tau(t)\}\). For any newly enabled transition \(t'\), the dynamic firing interval is reset to its static values: \(\overline{min}_t = \min_t\), and \(\overline{max}_t = \max_t\).

The state \(s_2\) with marking \(M_2\) is then said to be directly reachable from \(s_1\) with marking \(M_1\) by the firing of \(t\), denoted by \(M_1 \xrightarrow{t} M_2\). \(\{M\}\) denotes the set of markings reachable from \(M\). \(\rightarrow^*\) is the forward reachability relation, i.e., \(M \in \{M_1\} \iff M_1 \rightarrow^* M\).

Since TWF systems are 1-bounded by definition, safety definition in terms of boundedness is trivial. Therefore, to ensure safety is defined as to resolve all contact situations. A contact situation [8] is one in which the pre-conditions and the post-conditions of an event are satisfied at the same time, as shown in Figure 1. In untimed EN systems, the transition \(t_1\) with pre- and post-conditions marked at the same time cannot fire. We call transition \(t_1\) unsafe. In our study, we want to explicitly distinguish the order (intended or arbitrary) of events occurring (transitions firing) using time constraints - for instance, that \(t_2\) will definitely fire first before \(t_1\). Contact situations are permitted as long as tokens in the event's post-conditions are guaranteed to disappear before the event occurrence. In the workflow context, the execution of task \(t_2\) must be completed before task \(t_1\). This is, in fact, our notion of safety and time safety.

![Figure 1: A Contact Situation](image)

Given a TWF system \((N,M_0)\), a transition \(t\) is safe in state \((M, \overline{FL})\) iff \(*t \subseteq M\) or \(*t \cap M = \emptyset\). It is unsafe otherwise. It is timesafe if either \(t\) is safe; or for every \(t_i \neq t\) such that \(*t_i \subseteq M, \overline{FL}(t) \subseteq \overline{FL}(t_i)\). In other words, an event is safe if no contact situation can occur; or it is timesafe if the contact situation is resolved with the given time constraints. Hence, a TWF system is timesafe if and only if every transition is timesafe in every reachable state. Otherwise, it is unsafe.

In untimed Petri net systems, a transition is live, as long as it can be enabled in subsequent states. In TWF systems, the transition must not only be enabled, but also firable. As in previous work [13], we define liveness such that transitions must be firable in the future, whatever state the system is in.

A transition \(t\) is live iff for every reachable state \(s \in \{S_0\}\), \(t\) can fire in some state \(s'\) reachable from \(s\). The system \((N,M_0)\) is live if for all \(t \in T_N\), \(t\) is live. The transition \(t\) is dead iff \(t\) is not enabled in some state reachable from \(s\).

Like WF-nets, we define soundness for TWF-nets, whose initial marking must include the place \(i\). Let \(i \cup R\) be the initial marking, where \(i \notin R\) and \(o \notin R\). \(R \subseteq P_N\) represents the set of resources that already exist in the system. Then, a TWF-net \(N\) is sound iff (1) \(\forall M \subseteq P_N, \{\{i \cup R\} \rightarrow^* M : M \rightarrow^* \{\{o \cup R\}\}\), i.e., we can always reach the final state \(\{\{o \cup R\}\}\), at which we exit the workflow at \(o\) and restore the available resources \(R\); and (2) no transition \(t \in T_N\) is dead in \((N,R \cup \{i\})\).

The above covers the three workflow restrictions as imposed for untimed WF-nets [6]. In other words, regardless of whether it is timed or untimed, a case must have the option to complete, have a proper completion and there are no dead task. For TWF-nets, we also allow other tokens to exist in the system to represent...
resources.

Like WF-nets [15], we define an extended net $\overline{N} = (\overline{P}, \overline{T}, \overline{F}, \overline{FI})$ given a TWF-net $N = (P, T, F, FI)$ and an extra transition $t^* : P = P, \overline{T} = T \cup \{t^*\}, \overline{F} = F \cup \{(o, t^*), (t^*, i)\}$ and $\overline{FI} = FI \cup \{(t^* \rightarrow [0, 0])\}$. $t^*$ connects place $o$ to place $i$ and thus closes the system. $t^*$ caters for strong connectedness and is an immediate transition with no duration.

In our previous work [16], we modify some well-known free-choice Petri net theorems to cater for time intervals. The results were the liveness criteria for free-choice TEN systems and they were recalled below. The respective proofs of these theorems can be found in [16].

**Theorem 1**

A timesafe free-choice TEN system is live if and only if every proper siphon contains an initially marked trap.

**Theorem 2**

A timesafe free-choice TEN system is live if it is covered by strongly connected S-components and every minimal proper siphon is the set of places of a marked S-component.

Further, Theorem 1 can be generalised, as in [11] and applied to asymmetric-choice nets.

**Theorem 3**

A timesafe asymmetric-choice TEN system is live if every proper siphon contains an initially marked trap.

We also note that TWF systems are exactly the same as TEN systems and the above can be readily applied to TWF systems.

Next, we formulate a necessary and sufficient condition for the soundness of a timesafe TWF-net [16].

**Theorem 4**

A TWF-net $N$ is sound if and only if $(N, M_0)$, where $i \in M_0$, is live and $M_0$ is a home marking.

Comparing with the soundness condition for untimed WF-nets - A WF-net $N$ is sound if and only if $(N, \{i\})$ is bounded and live [15], Theorem 3 omits the boundedness property because the property is trivial in our system. In our previous work, we have also shown that liveness is closely related to safety [16].

The method used to check the soundness of a given TWF-net depends on the complexity of the net. We can use straightforward reachability analysis to ensure that the three requirements for soundness hold. If the system is too complex, other methods of analysis without the need to generate large state space can be used. These methods include the structure theory for free-choice Petri nets [11]. However, existing theorems need to be modified to incorporate our notion of time safety in the same fashion as Theorems 1, 2 and 3.

4 A Patient Workflow Management

To illustrate the notion of safety in TWF-nets, we use a workflow example described in the study on a Patient Workflow Management System (PWMS) [10] based on Guidelines for clinical practice. The case study described a generic health care guideline to manage Acute Myeloid Leukemia (AML) in children. The general guideline is reproduced as a WF-net shown in Figure 2.

![Figure 2: A Health Care Process in WF-net](image)

Each task in the WF-net can be further broken down into sub-tasks or elementary process steps [10]. Consider the task of PLT and PMN Count (Platelet and Polymorphonucleosytes Count). It can be broken down into three sequential sub-tasks in the following order: draw blood, count PLT and PMN and report results. Each sub-task requires a resource with a particular role description to perform the task [10]. For instance, the drawing of blood is performed by a nurse while the latter two tasks are performed by an analyst. The allocation of resources is controlled by a resource manager. The manager accepts a request and then assigns the appropriate resource when available.

Figure 3 shows a modified version of these ideas in Petri net. The top row depicts the sequential sub-tasks (part of the WF-net), while the rest illustrates the interaction with the resource manager. Before the drawing of blood, a nurse (resource) has to be requested and then assigned to draw the patient’s blood.

Now, consider two parallel processes with two patients awaiting the same procedure but with only one nurse available, as shown by the TWF-net $N$ in Figure 4. Both patients are competing for the same nurse by in-
teracting with the resource manager subnet. Based on our definition of safety, the untimed version is not safe. We add time constraints to provide time safety in this regard. Essentially, the time constraints ensure that Patient 1 is to put in a request, and thus complete the procedure first before Patient 2 does.

Figure 4: Two Patients and One Nurse

Clearly, the patient management system permits the modelling of shared resouces. Each patient is competing for the limited resource (nurse) required for the task in real time. Although in our example we model both threads performing the same sequential tasks, intuitively, resource sharing can also be modelled for different workflow processes. Unrelated workflow processes from different organisations can have access to shared resources.

5 Conclusion and Future Work

We have provided a time interval extension of WF-nets for the purpose of modelling and analysing workflow systems with time constraints. With this new extension, we have introduced a notion of safety in time which is analogous to the usual notion of boundedness in Petri nets. Using the new safety notion and after some minor modifications, we apply the new theorems to Time WF-nets. While the results are non-trivial, they basically preserve the soundness theorems of workflows and the liveness theorems of Petri nets in the combined time and WF-nets.

Existing classes of nets with time intervals on transitions are the Time Petri Net (TPN) model [17, 18] and the Interval Timed Coloured Petri Net (ITCPN) model [14]. We adopt the TPN semantic model in our work because we find the model corresponds more adequately to real-time and real-world modelling [13]. It is expected that existing verification methods based on state classes devised for TPN and ITCPN [18, 19] can be readily modified and applied to TWF systems for workflow analysis.

Duration constraints, same as time intervals, have already been assigned to individual workflow tasks from which temporal consistency of the workflow system can be verified [3]. Based on these duration constraints, algorithms to calculate the longest/shortest process instance in a workflow graph were formulated and various temporal requirements and inconsistencies can also be checked using verification algorithms. The duration constraint model is intuitively similar to the time interval notion of TWF-nets. Therefore, it appears that these algorithms can be readily applied to TWF-nets with minimal modifications. However, these approaches were formulated in terms of graphs and do not cater for the complexity of concurrent behaviours. They do not analyse deadlock freedom, liveness and fairness. Our methods allow the verification of such well-behavedness properties in the presence of real-time constraints.

TWF-nets allow multiple tokens in the system. This feature provides us with a richer semantics for the workflow model. Our extended net is able to model several processes, competing for available resources provided for the performance of a given task. To realistically model workflow in real time, it is important to consider multiple threads whose flows may overlap in time to compete for the limited resources. This feature can also be applied to the modelling of resource sharing in interorganisational workflows [20]. Rather than having resource sharing within a single workflow process as shown in our example, the resource manager can control the timely allocation of resources among a set of loosely coupled workflow processes from different organisations.

Currently, our study assumes safety to deduce system liveness. To produce a more complete and rigorous analysis method, a comprehensive (time) safety test on the system will need to be developed. Another natural progression of our current work is to include object-oriented features. One can consider each token in a TWF-net as an object instance identified by a case id. Different cases can then compete for shared resources available to complete the tasks in time.
References


