Timed Automata for Workflow Modeling and Analysis

Afsoon Soltani
Department of Computer Engineering, Arak Branch, Islamic Azad University,
Arak, Iran
Email: afsoon.soltani@yahoo.com

Abstract- The main purpose of the workflow management systems is supporting of definition, performing and controlling of the business process, so that it supports the activities and the particular ordering of them in order to achieve to the common goal. The accuracy of the characteristics of the workflow for automation of the Business Process is essential. Because of this reason, the errors in the characteristics should be known and corrected as soon as possible. Combination of the timed constraints in such systems deteriorates the issue. The unexpected delays may result in catastrophic destroying or deficiency in Business Activities. The workflow existent systems provide limited support of the time issues. However, the implementation of great unverified workflow models runs the risk of undesirable runtime. Using of the Formal Methods for automating verifications is very important and essential. But because of the intricate understanding of math, it is considered a little. So the existent solution in order to rich the workflow with one series of the timed constraints which are needed, combine the modeling with the formal comprehension of Timed Automata. This exact solution can help to guaranty the validity of the workflow systems for analyzing and verifying, use the UPPAAL tools.

Keywords : Workflow, Automation, Verification, Formal Method, Timed Automata

1. Introduction

Information technology has gone through one of the most significant changes in the last decades, meanwhile having proliferated and penetrated into business processes in each and every business field, and has become a determining component[1] . A workflow
is a collection of activities, agents, and dependencies between activities. Activities correspond to individual steps in a business process, agents are responsible for the enactment of activities, and dependencies determine the execution sequence of activities and the data flow between them. Time-related factors have to be incorporated into the traditional workflow processes so as to adapt to the globally distributed applications[2,3]. Time violations in workflows can often lead to some negative influences such as increasing cost, unexpected delays and resource wasting, or even cause catastrophic breakdowns within business processes. However, there exists no standard for the temporal behavior of workflow modeling. The correctness of a workflow specification is critical for the automation of business processes. For this reason, errors in the specification should be detected and corrected as early as possible at specification time. While formal verification methods can help ensure the reliability of workflow systems, the industrial uptake of such methods has been slow largely due to the effort involved in modeling and the memory required to verify complex systems. Timed automata are one of the most widely used formalisms for the specification and verification of real-time systems. Recently, a number of authors have advocated the use of Timed Automata as a semantic for modeling techniques in computer science[4]. What remains to do is to translate the workflow specification and the requirement we are interested in into the input language of a model checker. Model checking is an automatic analysis method, which explores all possible states of a modeled system to verify whether the system satisfies a formally specified property.

The present approach which enriches timed workflow is considered to be a new idea. Furthermore, Timed Automata has been applied to formalize workflow model as one of the most widely used methods in formalization [4]. The present approach enjoys high level of accuracy and is able to ensure the reliability of the workflow systems by using UPPAAL tools for verification and analysis.

The rest of the study is organized as follows: part 2 will concentrate on reviewing the relevant literature. The definitions of the key concepts are included in part 3. A modeling approach for timed workflow models is presented in part 4. Afterwards, part 5 will face with the verification and evaluation of the presented timed workflow model. Finally, the conclusion of the present study and strategies for future studies will be discussed in part 6.
2. Related Works

The area of handling time-related issues and detecting potential problems has not received adequate attention in the workflow literature. Timed Automata has been suggested as a tool for specification and verification of timed workflow schema [10]. The present approach has decreased the compatibility of workflow schema with the emptiness problem of timed Automata. A method of verification of the workflow's characteristics is presented [14] by using model checking. Having used timed Automata, the present approach has presented a solution for scheduling problem which is much easier than Resource Constraint Project Scheduling (RCPSP).

Timed Petri Nets [15,16] have recently turned into a well-known widely used model consisting of distributed systems and Timed Petri Nets are presented [18] for modeling the behavior of workflow systems by using Time Petri Net Analyzer (TINA) as a tool. This method is based on a united time graph. An automatic translator YAWL2DVEt is presented [19] where a timed workflow modeled by YAWL tools are the inputs. The automatic translator presents a method for verification of the timed workflow system. A slot has been assigned to each task in the workflow diagram[15]. The present study has tried to develop the workflows with slots which are going to be called Time WF_Net from now on. For the visualization of the time data of the workflow complex systems, a modeling method for workflow systems based on Time Constraint Petri Nets (TCPN) has been suggested [17]. The main idea which has been indicated [13] is to enrich a characteristic of workflow with time data for the activity and to transform this workflow description into PERT_Diagram. This approach has presented a PERT_Diagram based on technics to manage the activities within the framework of the definition of the process. As a result, with the goal of developing technics to control the time vulnerability, a framework is offered to calculate the deadlines of the activities [20].

Nowadays, the graph transformation systems have become well-known in defining models' formal concepts. An approach to define the formal definition of UML 2.0 Activity Diagrams has been introduced through graph transformation [21]. They have implemented AGG tools. Also a modeling approach toward workflow was presented by setting time constraints and distributing the activities based on Multiple Time Axes [22].
3. Basic Concepts

This section provides background information about the tools and concepts used in this work.

3.1 Workflow Model

A Workflow Management System (WFMS) is defined as a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic[2,4].

3.1.1 Routing in workflow

The routing describes the control-flow of the Workflow model. It is a formalized view which is present as a co-ordinated set of process activities that are connected in order to achieve a common goal [6,7].

Sequential Routing

the tasks have to be carried out sequentially, one after the other.(see Fig1)

Parallel Routing

we refer to parallel routing, when more than one process activities can be carried out at the same time (see Fig 2). To execute the parallel routing at the beginning we must use an AND-split to split the control and an AND-join to collect it at the end.
Selective Routing

we refer to selective routing, if only one of the process activities have to be carried out at the same time. To execute selective routing at the beginning we must use an OR-split to split the control and an OR-join to collect it at the end. The real function of the OR-join is to choose between the given possible process activities. (See Fig 3).

![Selective Routing](image)

3.1.2 Structural Correctness in workflow schema

A workflow specification is structurally correct if every workflow execution reaches the end node after a finite number of transitions. And when the end node is reached, all other activities that have been started before are completed and there are no remaining join nodes waiting for incoming transitions. An example is shown in Fig 4 Only one of the activities 2 and 3 will be performed after the OR-split, but the following AND-join would wait for both activities being completed. For this reason it is reasonable to call such a workflow specification structurally incorrect.

![Incorrect Workflow](image)
3.1.3 Timed Workflow Schema

At Fig 5 the workflow is showing the "sell computer hardware: business process of an E-Business company which sells computer hardware products. When the hardware store receives an order from a customer we begin at initial node action of the diagram. After the action is done we do the "check order" action. When the order is checked there is a And_Split. Which means that the "get products" action and "save order information in archive" action are performed at the same time and the sequence between them is irrelevant. After the "save order information" action id done there is an And_Join. This means that the "assemble bundle" action is not performed before both incoming actions are done. After the "get products" action there is a OR_Split node. If the product type is a computer the "test computer" action is performed, if it is a monitor the "test monitor" action is performed. Now the second incoming action of the "assemble bundle" action is done and this satisfies the And_Join control structure which means that the "assemble bundle" action can now be performed. The time units are meant to be minutes but we will not go into detail with our choice of numbers since it is irrelevant how realistic the time intervals are. For example the [5,15] above the "get products" action means that this action can take between 5 and 15 minutes to complete.

3.2 Timed Automata

Timed automata are one of the most widely used formalisms for the specification and verification of real-time systems[9,11].

922
Definition 1

A timed automaton \( A \) is a tuple \(<S, S_0, \Sigma, C, I, E, F>\), where \( S \) is a finite set of locations, \( S_0 \subseteq S \) is a set of initial locations, \( \Sigma \) is an input alphabet, \( C \) is a finite set of clocks, \( I \) is a mapping from \( S \) to a set of clock constraints, \( E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(c) \) is a set of switches, \( F \subseteq S \) is a set of final locations. For any location \( s \), \( I \) specifies the time constraints that must be satisfied to remain in \( s \) (invariant set). As soon as an invariant is violated due to the elapse of time, we must exit the location. \(<S, S', a, \lambda, \delta>\) represents a transition from location \( s \) to location \( s_0 \) on input symbol \( a \); the set \( \lambda \subseteq C \) specifies the clocks to be reset by the transition, while \( \delta \) is a clock constraint over \( C \) that specifies when the switch is enabled [10,23].

3.3 Model Checker UPPAAL

UPPAAL is a toolbox for verification of real-time systems. The model-checker UPPAAL is based on the theory of timed automata [24] and its modeling language offers additional features such as bounded integer variables and urgency. The query language of UPPAAL, used to specify properties to be checked, is a subset of CTL (computation tree logic) [25]. The query language consists of path formulae and state formulae. State formulae describe individual states states, whereas path formulae quantify over paths or traces of the model. Properties that could be checked include:

Dead Lock Freeness Property

A state is a deadlock state if there are no outgoing action transitions neither from the state itself or any of its delay successors[12].

Reachability Property

They ask whether a given state formula, \( \phi \), possibly can be satisfied by any reachable state. In UPPAAL, we write this property using the syntax \( E<> \phi \).
Safety Property

Safety properties are on the form: “something bad will never happen”. In UPPAAL we write $A[\emptyset]$ and $E[\emptyset]$, respectively.

Liveness Property

Something will eventually happen. liveness is expressed with the path formula $A^0 \emptyset$, meaning $\emptyset$ is eventually satisfied.

4. Timed workflow modeling

Using the elements introduced in the last section, we can define templates for the different kinds of nodes in a workflow specification. The present study has used token concept for modeling. At the beginning there is only one token in the workflow and that belongs to the Start Node. Each node runs only when it is accompanied by a token, then it will pass the token to the next node. The token will disappear when it reaches the final token at last.

4.1 Global declaration

Regarding the fact that a workflow model is consisting of a number of nodes and a model may contain several instances of each of the nodes, therefore, a constant has been defined for each node. Thereafter, the numbers of each node have been limited by defining the data type and with regard to the variables of the same type.

\begin{verbatim}
const int Act;
typedef int[0,Act-1] act_id;
\end{verbatim}

Array variable t is used for keeping time of each task implementation and totaltime variable is used for keeping the global time. In order to synchronize the nodes, the broadcast channel has been defined.

\begin{verbatim}
clock t[Act];
clock totaltime;
const int MaxSizeNode;
broadcast chan c [MaxSizeNode];
\end{verbatim}
Each node keeps the required data in a structure data type. Only one instance of the structures has been mentioned due to the similarities between the nodes.

```c
typedef struct {
    int PostTyp;
    int PostID;
    bool HasToken;
    bool Visited;
} StartNode;
StartNode Start;
```

Given the fact that each node may contain several instances; an array from each structure has been taken into the consideration.

### 4.2 Workflow Elements

Using the elements introduced in the last section, we can define templates for the different kinds of nodes in a workflow specification.

#### 4.2.1 Start Node Template

After the condition of possessing the token became true, the Start node which has received a C signal, will pass it to the next node. Then, token status will be marked false but its own visited status will be marked true. Finally, in order to synchronize, it will send a C signal on the channel to the next node. (See Fig. 6)

![Fig.6. Start Node](image)

As an example, the SetStartToken function changes HasToken field of the next node's to True.

```c
void SetStartToken(bool b){Start.HasToken=b;}
```
4.2.2 Activity Node Template

Like the Start Node, this process has to attend the activation. Then, variable t should be set on zero and transfer the token to the next node. Working state, the Process rests at the minimum time and maximum time. Then it marks the visited characteristic and the next node's token true and its own token false. Finally, it sends a signal to the next node. (See Fig.7)

\[c[id]?, t[id] = 0, \text{TokenAct(id)}\rightarrow \begin{cases} \text{working, t[id] \geq \text{MinTime(id)}} \\ \text{final, c[NextAct(id)]=1} \end{cases} \]

Fig.7. Activity Node

4.2.3 And_Split Node Template

This process will simultaneously activate more nodes along the next one, therefore, the signal will be sent twice and the token of the next two nodes will be marked True. (See Fig.8)

\[\text{SetTruTokNxt1AS(id)} \rightarrow \text{c[Next1AndSplit(id)]!, TokenAndSplit(id), c[id]?, FalseTokenAndSplit(id), SetTrueTokenNxt2AndSplit(id)} \rightarrow \text{c[Next2AndSplit(id)]!, TrueVisitAndSplit(id)} \]

Fig.8. And_Split Node

4.2.4 And_Join Node Template

And_Join starts when both of the pervious nodes had been visited. In that case the process will move to the next state. When the token of the next node is activated, the
status of the next token and the status of the visited one will be marked subsequently False and True. Consequently a signal will be sent to the next node. (See Fig.9)

4.2.5 OR_Split Node Template

Unlike And_Split, the OR_Split process will only activate the next node. Thus, after changing the status of its characteristic, it will change the status of one token of the next nodes to True and will send it a C signal through synchronization channel. (See Fig.10)

4.2.6 OR_Join Node Template

OR_Join will start when not only when C signal is received and the status of the token is True, but also when it finds one of the pervious nodes visited. In that case, the token and the visited one will change their statuses, mark the next node's token as True, and send a signal on Channel C for the next node. (See Fig.11)
4.2.7 End Node Template

While this process is activated, the token characteristic will be marked false and it will enter the terminal state. From this stage on, the available token will disappear in the workflow and none of the nodes will be working. (See Fig. 12)

5. Verification and Evaluation of the Timed Workflow Model

In this section we are going to use UPPAAL to verify the most interesting properties of the system we modeled.
Property 1) The terminal Node will be available in all routes. 
\[A<> \text{End.terminal}\]
This characteristic which verifies the structure correctness will be evaluated True for the given system. However, if OR_Join and And_Join change their places, this characteristic will be overturned. The tools will identify the error position by trying a counterexample.

Property 2) There is no deadlock in the system. 
\[E<> \text{deadlock}\]
This characteristic was evaluated to be True. This will seem to be surprising at first but closer observation will lead us to admit that the system will always end up in the terminal state. Otherwise stating, the system always ends up in deadlock.

Property 3) Test monitor will finally be carried out. 
\[E<> \text{Activity(4).final}\]
This characteristic was evaluated to be True. We will study a characteristic here which is apparently incorrect.

Property 4) Both monitor test and computer tests activities could be implemented. 
\[E<> \text{Activity(3).final + Activity(4).final >1}\]
We are directed toward a counterexample through not only the receiving tools but also by the overturned characteristic. The OR_Split node will only result in the performance of the one of the next nodes. Overturn of this characteristic is obvious.

Property 5) Finally the data will be saved in the archive if check order activity is carried out in the system. 
\[\text{Activity(0).final} \rightarrow \text{Activity(5).final}\]
This characteristic was evaluated to be True.

Property 6) Check order activity will always precede get product activity. 
\[\text{Activity(2).working} \rightarrow \text{Activity(0).final}\]
This characteristic was evaluated to be True.

**Property 7)** Minimum time of the workflow

\( \text{E} \rightarrow \text{End. terminal} \)

To study the minimum time of the workflow, the tracking feature of the diagnostic tool has been used to analyze the total time. The tool showed total time \( \geq 15 \) which is the minimum time of the workflow. However, to verify the accuracy of the tool, the following characteristic was checked.

**Property 8)** Workflow could be completed in less than 15 minutes.

\( \text{A} \rightarrow \text{End. terminal and total time}<15 \)

This characteristic was evaluated to be False which confirms the correctness of the previous characteristic. But in more complex systems with a lot of splits and joins it will be very complicated to do in that way. So in these cases UPPAAL can be a great help.

Model checking will study all possible states of the modeled system. One of the critical issues of all model-checking approaches is the memory limitation problem. The amount of memory used by the approach has been evaluated in Table 1.

<table>
<thead>
<tr>
<th>Property</th>
<th>True/False</th>
<th>Memory(KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
<td>9,488</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td>8,980</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
<td>9,492</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>8,296</td>
</tr>
<tr>
<td>5</td>
<td>True</td>
<td>9,304</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>8,844</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>10,280</td>
</tr>
<tr>
<td>8</td>
<td>False</td>
<td>9,512</td>
</tr>
</tbody>
</table>

The results of the study indicated that this approach would use a small size of the memory for the verification, and thus, could increase the reliability of the workflow systems.
6. Conclusion

The objective of the present approach is to enrich the workflow with required templates for the current applications. In order to model and verify, this approach has been concentrating on accuracy and validity issues by using timed Automata as an accurate formal concept in specification of the concepts. Moreover, to verify the characteristics of checking tools, tool checker of the UPPAAL model has been applied where one characteristic of workflow and the requirements of the correctness have been translated to state machine. Therefore, the requirements to design workflow system with high levels of reliability have been meet. Using only one tool for verification of different characteristics and easy translation of UPPAAL to other characteristics is one of the privileges of this approach. The findings of the study confirm that this approach will occupy a small size of the memory.

7. References


