Petri net based Verification of a Cooperative Work flow Model

Annappa B, Jiju P, K Chandrasekaran, K C Shet
Department of Computer Engineering, National Institute of Technology, Karnataka, India
annappa@ieee.org, jijupnair2000@yahoo.co.in, kchnitk@ieee.org, kcshet@rediffmail.com

Abstract

This paper exploits the theory of Petri nets to verify reachability and soundness of a cooperative workflow model. First, we outline a cooperative workflow model, which is a modified version of Bonita workflow model. Bonita is open source cooperative workflow management software which is an ongoing project from object web consortium. Then we describe the cooperative workflow model using a special kind of Petri net called Wf-net. Next we employ WF-net for verification of the model for reachability and soundness properties. The Petri net based verification shows that the model is reachable and sound.

1. Introduction

Workflow Management Systems have become a popular tool to support logistics of business processes in various environments like insurance, banking, office administration, industrial manufacturing etc. Nowadays, Business organizations use workflow technology for the automation of their business processes because of its effectiveness in improving the overall throughput of the organization and saving the cost of the work. Once, Workflow management System has been adopted by any organization, the managers can focus on staff and business issues such as individual performance and optimal procedures rather than the routine assignments of tasks. Moreover there will not be any misplacement of work as the flow of work is fully automated. Currently many vendors are offering the workflow management system, which clearly shows that the software industry has already recognized the potential of workflow management tools.

A workflow model is a set of activities which are abstracted from actual business process [1]. It is often used to define concrete workflow. Workflow models can be broadly classified as traditional workflow models and cooperative workflow models. The main difference between the two is the way in which the activities in the workflow share their results. In cooperative workflows, activities can share their intermediate results not only at the beginning and end of the execution but also in the middle of the execution whereas in traditional workflows activities can share their results only at the beginning and end of the execution. As nowadays virtual enterprises have become a common place on the web, there has been a paradigm shift from the traditional workflows to cooperative workflows. The concept of virtual enterprise depicts the idea that many applications are the result of cross-organizational cooperation between several actors, playing different roles, who build a relational system which is structured by a common objective. Very few attempts have been made to design a cooperative workflow model till date. Most important of them is Bonita workflow model from Object web consortium. We have modified Bonita workflow model slightly to incorporate COO transaction protocol [2] and an efficient exception handling mechanism. The modified version of the Bonita is described in the section II of this paper.

We use Petri nets to describe the entire model so that verification of different properties of workflow can be done with the help of Petri net theory. It is important to note that Petri net has been chosen to verify the model though there are variety of other tools including UML diagrams, state charts and labeled transition systems. This is mainly because it has a strong mathematical foundation with various mathematical properties like reachability, liveness and boundedness.

The area of Petri nets was initiated by C.A. Petri in 1960's. Petri nets are used to model business processes, conceptually and mathematically in a very effective manner. In [3], W.M.P Van der Aalst projects three good reasons for using Petri net based Workflow Management Systems. First one is its ability to represent business logic in a formal and graphical way. In contrast with many other process modeling techniques, the state of a case can be modeled explicitly in a Petri net. Above all these, Petri nets are blessed with the abundance of analysis techniques. In this paper we use WF-net, a special kind of Petri nets to describe the workflow model. Further, verification for two important properties, reachability and soundness of WF-net has been carried out using reachability graph.

Rest of this paper is organized as follows. Description of the cooperative workflow model is given in the section II. Some of the basic

978-1-4244-4615-5/09/$25.00 ©2009 IEEE

terminology related to Petri nets is explained in the section III. Section IV details the WF-net description of the model. Sections V and VI explains the reachability analysis and soundness analysis of the model respectively. A conclusion of the paper is given in the section VII followed by references.

2. Description of the cooperative workflow model

In this section, we describe a cooperative workflow model which is a modified version of Bonita [4], [5] workflow model. A Cooperative workflow model allows the activities to share their intermediate results not only at the beginning or end, but also in the middle of execution.

The new cooperative model is designed to address some of the limitations of the Bonita Workflow management system. So the main intention is to modify the Bonita Workflow model so as to incorporate an efficient exception handling mechanism, cooperation of the intra or inter organizational processes/activities and an effective transaction protocol. By including the transaction protocol and cooperation mechanism, the Bonita can be extended to use in collaborative applications in inter organizational environment. Here, transaction protocol ensures the correctness and consistency of the application [6].

The model consists of various stages (states) in the life cycle of an activity in the workflow. It also depicts the events which triggers the transition from one state to another state of an activity. The figure 1 shows the modified version of the Bonita workflow model [7].The new states added to the model are *Communicating*, *Pause*, *Handling Exceptions* and *Ready to complete* (RTC).

Different states of the activity are explained in the following.
Initial: This is the state of an activity waiting for some processing to be completed before being ready to run. In the case of normal activities, at least one of the parent activities is still executing. In the case of anticipatable activities, if at least one of the parent activities has not yet started.
Ready: This is the state of an activity ready to be started. There are two possible situations for this state to occur. In the first, an activity has no parent activity (this is the first activity of the workflow process). In the second, a normal activity has parent activities that have all terminated successfully. Besides this, the transition conditions of the parent activity to this activity have been successfully met.
Executing: A state where activity executes predefined actions.
Anticipatable: This is the state of an activity that can be started without waiting for its parent activity(s) to complete. However, all of the parent activities must have started.
Anticipating: A previously anticipatable activity that has been started. However an anticipating activity cannot be terminated until all parent activities have terminated and the transition conditions have been successfully evaluated.
Terminated: A cancelled activity. This occurs in two situations. An activity can be explicitly cancelled or an unsuccessful evaluation of an inner transition condition.
Completed: An activity that has completed successfully.

Communicating: The activities go to this state when they want to access any resource from other activities or when they receive a request for its own resource from other activities. The resource may be the intermediary results of the activities.

Another instance where the activity can go to the communicate state is when the activity ask for any updates in other parallel activities on which it is dependent. Once the activity goes to communicating state, it may return to executing state, provided the requested resource is sent or received or the updates are complete. The dependent activities which have already produced its final results can go to Ready to complete (RTC) state if its partner activity in the dependency relationship is still executing.

Ready to Terminate: This state has been added to the model to incorporate the COO transaction protocol [2], [6] into the workflow which forms the foundation for cooperation among the activities in the inter organizational environment.Inter organizational collaborative workflows consists of various business processes from same organization or different organizations which participate in an application to achieve a common goal. Business processes consist of activities which are indivisible.

In COO-transaction protocol, if an activity uses an intermediate result of another activity, then a dependence relationship of the two processes involved is created. This dependence relationship will be removed only when the dependent process uses the final result of the other process. The significance of Ready to complete (RTC) state arises when the dependence relationship among various processes form cycles. Such cycles are referred to as groups. In this case, transaction protocol allows the group termination process only if all the process in a group are in Ready to complete state.

As already mentioned in the communicating state, an activity goes to RTC state when it completes its execution and produces final results but its partner activity in the dependency relationship is still running. Because of the dependency the activity cannot be terminated.

Paused: This is a state of an activity where user explicitly pauses a running activity for some reasons. For e.g., In Loan request process workflow which consists of various activities like *loan search*, *loan request*, *preliminary validation*, *financial check*, *job check*, *final approval* and *loan acceptance*, if a bank employee wants to attend a higher priority work for a while when he was on the *financial check* page of the workflow, he can explicitly pause that application and attend the other work. Once he finishes the higher priority work, he can resume at the point where he left in *financial check* activity of Loan request process. So this state gives flexibility to the workflow system. The activity will resume execution when user does the appropriate action. An activity in the pause state will be terminated after a specified time interval.

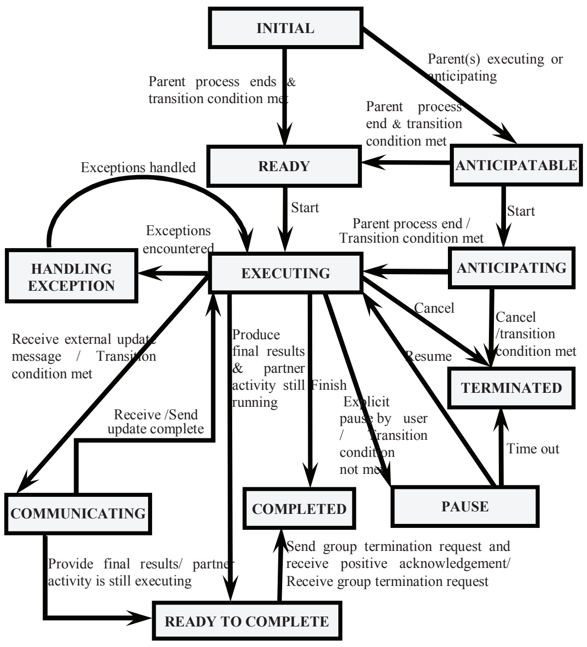
INITIAL
Parent(s) executing or anticipating
Parent process ends & transition condition met
Parent process end & transition condition met
Exceptions handled
READY
ANTICIPATABLE
Start
Start
Exceptions encountered
Parent process end / Transition condition met
HANDLING EXCEPTION
EXECUTING
ANTICIPATING
Cancel
Cancel /transition condition met
Receive external update message / Transition condition met
Produce final results & partner activity still running
Finish
Resume
TERMINATED
Explicit pause by user / Transition condition not met
Receive /Send update complete
Time out
COMMUNICATING
COMPLETED
PAUSE
Send group termination request and receive positive acknowledgement/ Receive group termination request
Provide final results/ partner activity is still executing
READY TO COMPLETE

Figure 1. Proposed Modification for the Bonita

Handling Exceptions: One feature that is lacked in the Bonita workflow model is an efficient exception handling mechanism. So in this model, exception handling is made as a different state of an activity where the activity will be dealing with the exception handling policies and routines.
An executing activity will go to handling exception state when any of the exceptions are encountered. Exceptions may be of the kind behavioral, semantic or system based. Once the exceptions are handled, activity goes to running state.
Different exceptions that may arise in the inter organization workflow management system can be broadly categorized into the following three categories [8].
*Behavioral Exceptions***:** This kind of exceptions is driven by improper execution order of the process activities. An example includes a user who uses the *back* button of a browser based application visits older page of the current activity, thus repeating that part of the activity.
*Semantic Exceptions***:** Semantic exceptions happen due to logical errors in the activities.
*System Exceptions***:** System exceptions are caused by the malfunctioning of the workflow based application infrastructure either at server or at client. These exceptions include network failures, system breakdown, data storage breakdowns and browser crashes. On the occurrence of an exception, the user should be notified about the exception and the exception should be resolved using the recovery policy. Exception handling mechanism consists of recovery operations that take place on the affected activities in order to bring the application into a consistent state, so that the process execution can proceed.

3. Basic terminology

Before going to the description of the workflow model using WF-net, we would like to define some of the terms which will make rest of the paper easy to understand.
A **Petri net** [9] is a tuple $N = (P, T, F, W)$ where:
1. $P \neq \varphi$ is a finite set of node places;
2. $T \neq \varphi$ is a finite set of node transitions;
3. $F \subseteq P \times T \cup T \times P$ is the flow relation;
4. $W: F \rightarrow \mathbb{N} \wedge [W(x, y) = 0 \Leftrightarrow (x, y) \notin F]$ is the weight function.

A Petri net which maps control flow dimensions of a workflow is known as **Workflow net** [9] (WF-net). It can be formally defined as below.
A Petri net $N = (P, T, F, W)$ is called a workflow net (WF-net) if and only if:
1. N has one source place i (i.e. $\bullet i = \varphi$) called initial place.
2. N has one sink place f (i.e. $f \bullet = \varphi$) called final place.
3. For every node $n \in P \cup T$, there exists a path from i to n and a path from n to f.

The state of a Petri net is determined by the distribution of tokens amongst the places. If workflows are mapped onto Petri nets, the state of a case will correspond to one or more tokens.

4. Description of the model using WF-NET

In this section, description of cooperative workflow model using WF-Net, a special kind of Petri net is illustrated. As illustrated in the figure 2, it consists of

9 places (c_i) and 14 transitions apart from start (source) and end (sink) places. The transitions are various triggers upon which an activity goes from one place to another. A place in this model is regarded as a state of an activity. The WF-Net in figure 2 treats both terminated and completed states of cooperative workflow model as sink place with unsuccessful and successful transitions respectively.

When the activity is started, it initializes its various resources and goes to initial state (c1). From initial state, it's an explicit-OR transition, which takes the activity to either anticipatable (c2) state or ready (c4) state. In ready state, the transition *execute* is triggered and the activity will go to running (c5) state. If the activity is in anticipatable state, it goes to anticipating (c3) state on the occurrence of *anticipate* transition. An activity in anticipating state may either go to running state if *its parent process ends* or to terminated (end) state on a *cancel*. A running activity has five alternative places to which it may go. These include the places *end, c6, c7, c8* and *c9*.

But here the transition from running to any of these five places may occur depending on the first transition which will be triggered. That is the reason why an implicit-OR is used after the running (c5) state. An activity in running state may be explicitly paused by a user and hence an arrow is used above the pause transition to show that it's a transition triggered by user. A paused activity may go to terminated (end) state on a *time out* or it may go to running state on *resume* transition. Since one transition is a timed trigger, an implicit-OR is used here also.

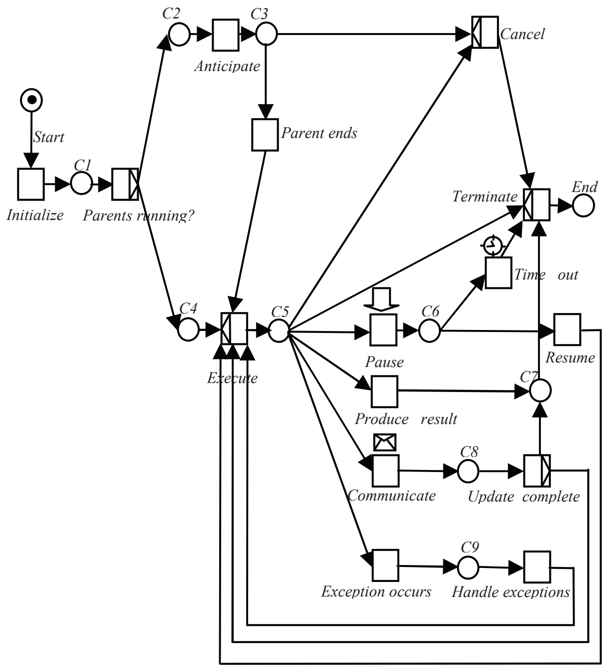

Figure 2. Petri net based description of the cooperative model

A running activity can go to RTT (c7) state if *it produces* final result and its partner activity in group with regard to the transaction protocol, is still running. An external update message can change a running activity to communicating activity on triggering *communicate* transition. Here communicate will be triggered using an external update message from another activity also. So it's a kind of external trigger and hence an envelope is used above the transition. After the completion of updates, communicating activity may go to either running state or RTT (Ready to terminate) state. If an *exception occurs* in an activity while running it goes to exception handling state (c9) and once the *exceptions are handled* it comes back to the point where it left in its running state when exception occurred.

5. Reachability Analysis

A firing sequence $w = t_1...t_n \in T^*$ can fire from m_0 to m_n, denoted by m_0 [w>m_n, if m_1, ...m_{n-1} exist with m_0[t_1>m_1[t_2>... [t_n>m_n where m_i are markings/places and t_i are transitions in the Petri net. The reachability problem is to decide for a given Petri net N, start- and end markings m_0, $m_e \in P$ if there is a $w \in T^*$ with m_0[w>m_e.

As far as the WF-net in figure 2 is concerned, the reachability problem can be defined as to check whether there exists a firing sequence $w=t_1….t_n \in T^*$ which can fire from *start* to *end* denoted by *start* [w>*end*. In the WF-net in the figure 2, any of the following firing sequences can fire from *start* to *end.*

1. Start>c1>c2>c3>end.
2. Start>c1>c2>c3>c5>end.
3. Start>c1>c2>c3>c5>c6>end.
4. Start>c1>c2>c3>c5>c7>end.
5. Start>c1>c4>c5>end.
6. Start>c1>c4>c5>c6>end.
7. Start>c1>c4>c5>c7>end.
8. Start>c1>c4>c5>c6>c5>end.
9. Start>c1>c4>c5>c8>c7>end.
10. Start>c1>c4>c5>c8>c5>end.
11. Start>c1>c4>c5>c9>c5>end.

There are more firing sequences which fires a token from start place to end place in the WF-net shown in figure 2. We have used the notion of reachability graph further to analyze the reachability behavior of WF-net. Reachability graph [10] is a directed graph consisting of nodes and directed arrows. Each node represents a reachable state and each arrow a possible change of state.

The reachability graph of WF-net in figure 2 is depicted in figure 3. The possible states of this network are indicated using a tuple (start, c1, c2, c3, c4, c5, c6, c7, c8, c9, end), with number of tokens in the various places. The initial state is illustrated as (1,0,0,0,0,0,0,0,0,0,0) where there is one token in start place. Different possible states are represented in reachability graph given in figure 3. Using the graph in figure 3; we can establish that for an initial WF-net with only one token, there are a total of 11 attainable states. Here each attainable state may not be occurred. A node with no arrows leading from it corresponds to an end state. In end state no transition is enabled. So if WF-net beginning with state (1,0,0,0,0,0,0,0,0,0,0) always has to result in end state (0,0,0,0,0,0,0,0,0,0,1). This is shown in the figure 3.

Another situation which contribute to the reachability in Petri nets are absence of deadlocks, live locks and dead transitions. These are generally occurred when AND-join and AND-split are present in the Petri nets. Since WF-net in figure 3 doesn't contain AND-Split, AND-join or a transition from which no leading arrows are present, the possibility of these three is ruled out. Hence reachability of WF-net given in figure 3 can be assured.

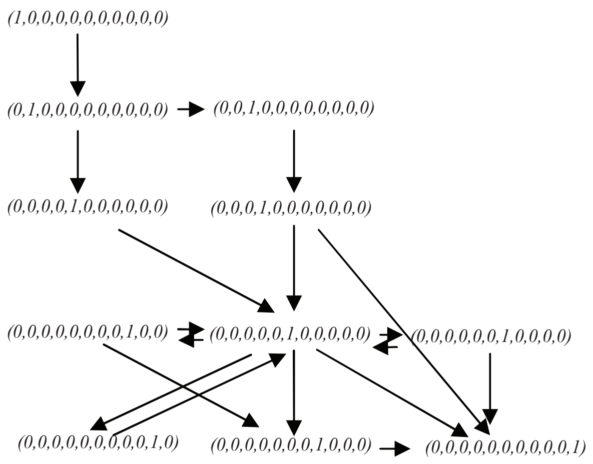

Figure 3. Reachability graph of WF-net

6. Soundness of WF-NET

A workflow net is sound if, and only if, it fulfills the following requirements:

1. For each token put in the place start, one (and only one) token eventually appears in the place end.
2. When the token appears in the place end, all the other places are empty.
3. For each transition (task), it is possible to move from the initial state to a state in which that transition is enabled.

A WF-net is defined to be sound if and only if, it is *live* and *bounded*. A Petri net is *live* when it is possible to reach, for each transition *t* and from every state reachable from the initial one, a state in which transition *t* is enabled. A Petri net is *bounded* when there is an upper limit to the number of tokens in each place.

Three soundness requirements of WF-net can be checked using reachability graph starting with the initial state in which there is only one token in the place start as shown in
figure 3. The first two requirements are checked by confirming that the reachability graph has only one final state, and that this is one in which there is precisely one token in *end*. Both of these conditions are satisfied in the figure 3. Last requirement is also verified by examining whether for each task, there is a state transition in the reachability graph which corresponds to the firing of that task (transition). As after analysis, we found that this requirement is also satisfied by WF-net, in figure 3. Hence we can conclude that proposed cooperative workflow model is sound.

7. Conclusion

Even though there are various tools for modeling and analyzing workflows, the most widely used one in the literature is Petri nets. This is mainly because of its extensive theoretical support. Moreover it avoids fuzziness and illogicality of the context being analyzed. Here we employ WF-nets for the verification of reachability and soundness of a cooperative workflow model. As far as the reachability of the model is concerned, it is shown that any token submitted to the WF-net ultimately reaches its end state. This implies that WF-net model and hence the cooperative workflow model is free from livelocks, deadlocks and dead transitions. Thus smooth state changes of an activity in the proposed cooperative workflow model are assured.

8. References

[1] Peng Liu, Tingting Fu, Ming Cai, Huaidong Shi, "An Improved Cooperative Model for Web Service Based Workflow Management", *Proceedings of the 11th IEEE International Conference on Computer Supported Cooperative Work in Design*, 26-28 April, 2007,pp. 818-822.

[2] C. godart, F. Charoy, O. Perrin and H. Skaf-Molli , "Cooperative workflows to coordinate asynchronous cooperative applications in a simple way", *Proceedings of Seventh IEEE International Conference on Parallel and Distributed Systems*, 4-7 July, 2000, pp. 409-416.

[3] W.M.P Van der Aalst, "Three Good Reasons for using a Petri net based Workflow management Systems", *Department of Mathematics and Computing Science, Eindhoven University of Technology.*

[4] Bonita. http://bonita.objectweb.org/

[5] Bonita Workflow: "Bonita API", Reference guide of Bonita v3.1, September, 2007.

[6] G. Canals, C. Godart, P. Molli, and M. Munier, "A Criterion to Enforce Correctness of Indirectly Cooperating Applications," *Information Sciences,* vol. 110/3-4, September, 1998, pp. 279- 302.

[7] Francois Charoy, Adnene Guabtini and Miguel Faura, "A Dynamic Workflow Management System for Coordination of Cooperative Activities", 2000.

[8] Marco Brambilla, Stefano Ceri, Sara Comai, Christina Tziviskou,"Exception Handling in Workflow driven Web Applications", *14th ACM International conference on World Wide Web,* ACM(2005) , pp.170-179.

[9] Kamel Barkaoui, Rahma Ben Ayed, and Zohra sbai, "Workflow Soundness Verification based on Structure Theory of Petri nets", *International Journal of Computing & Information Sciences, Vol. 5, No.1,* April, 2007.

[10] Wil van der Aalst and Kees van Hee, "*Workflow Management: Models, Methods and Systems*", ISBN 0262022891, MIT Press, December, 2000.

[11] WFMC, The Workflow Reference Model, Document Number WFMC-TC-1003, 1995.

[12] W. Aalst, "Modeling and Analyzing Inter organizational Workflows", *Proceedings of International Conference on Applications of Concurrency to System Design*, Fukushima, Japan, March 1998, pp. 262-272.