An Approach to Workflow Modeling and Analysis

Hemant Kr. Meena, Indradeep Saha, Koushik Kr. Mondal, T.V. Prabhakar
Dept of Computer Science and Engineering
IIT Kanpur
tvp@iitk.ac.in

ABSTRACT
In this paper we present a new approach to workflow analysis. We model the workflow using Activity diagrams, convert the Activity diagrams to Petri nets and use the theoretical results in the Petri nets domain to analyze the equivalent Petri nets and infer properties of the original workflow. We have demonstrated the possibility by developing an Eclipse plug-in, which can be used to model workflows using Activity Diagrams and then analyze these workflow models using Petri nets.

Categories and Subject Descriptors
D.2.2[Software Engineering]: Design Tools and Techniques - *Computer-aided software engineering (CASE), Petri Nets*

General Terms
Design

Keywords
workflow, activity diagrams, Petri nets, Eclipse, workflow analysis.

1. INTRODUCTION

Workflow modeling is an important phase in automating a process. However, modeling alone is not sufficient, as one should able to comment on certain properties of the given model. In this paper we present an approach to workflow analysis using Petri Nets. We first introduce workflow, Petri nets, and Eclipse in sections 2, 3, and 4 respectively. We then present our work in section 5. We propose to use activity diagrams for workflow modeling and then use Petri nets to analyze the workflow so modeled. Our work demonstrates that properties of a workflow can be inferred from its corresponding Petri net model. We have also built a plug-in on Eclipse [17], which provides an editor for workflow modeling using activity diagram. We then analyze this diagram by converting it into a corresponding Petri net model. We end this paper with some screenshots of our Eclipse plug-in and reviewing some related work in section 6 and 7 respectively.

2. WORKFLOW

Workflow refers to automation of business processes, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [5]. A workflow management system (WFMS) is a software package that is used to define, create and manage the execution of workflows. While designing a workflow, one describes which tasks have to be done and in what order. So process approach is given more importance. Hence it is important that a good modeling language is used to design a workflow.

OOPSLA, October 16–17, 2005, Saniego, CA, USA.

3. PETRI NETS

A Petri net is a directed graph with two types of nodes called places and transitions. An arc connects two nodes. A connection can be from a place to a transition or from a transition to a place.

A formal definition of Petri nets is as follows [2]:

A Petri Net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:

$P = \{p_1, p_2, p_3 \ldots p_m\}$ *is a finite set of places,*

$T = \{t_1, t_2, t_3 \ldots t_n\}$ *is a finite set of transitions,*

$F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs,*

$W: F \rightarrow \{1, 2, 3\ldots\ldots\}$ *is a weight function,*

$M_0: P \rightarrow \{0, 1, 2, 3\ldots\ldots\}$ *is the initial marking,*

$P \cap T = \Phi$ *and* $P \cup T \neq \Phi$

Marking denotes initial distribution of tokens among places. A transition is said to be enabled if each of its input place contains at least that number of tokens which is equal to arc joining the place and transition. An enabled transition may fire i.e. tokens are removed from its input places and added to output places.

While representing graphically, places are drawn as circles, transitions as rectangles, tokens as black dots, and arcs as arrows.

Here we present some useful properties of Petri nets:

Reachability: We start with some initial distribution of tokens among places, which we call initial marking of the given Petri net. When an enabled transition fires, the distribution of tokens change. Starting with an initial marking we can construct a reachability tree, which will produce all possible reachable markings. Since a marking represents a state in Petri net, from a reachability tree we can find out all possible reachable states of the given Petri net.

Coverability: Given a Petri net with initial marking M_0, a reachable marking M_1 is said to be coverable if there exist another marking M_2 whose distribution of tokens among places is either greater or equal to that of M_1.

Boundedness: A given Petri net with initial marking M_0 is said to be bounded if for any reachable marking, the number of tokens in each place does not exceed a finite value.

Safeness: A given Petri net with initial marking M_0 is safe, if it is bounded and maximum allowable token in each place is 1.

Liveness: A given Petri net with initial marking is said to be live, if from any reachable marking it is possible to fire any transition after some firing sequence. A transition t is said to be dead, if it can never be fired. If in a firing sequence we reach a point where a particular transition cannot be fired, then the net is in a potential deadlock.

4. ECLIPSE

Eclipse is a new way of looking at how we build tools, developed with a new vision -- a platform-centric rather than a tool-centric way of thinking. The bare bone Eclipse Platform is a "Universal IDE" -- an IDE for anything and nothing in particular [16]. Eclipse can provide meaningful integration across several tool stacks, which was hitherto impossible. Thus when we write our own tool plug-in which hooks into well-defined plug-in points in the Eclipse platform we teach Eclipse to solve a new problem rather than bolting a monolithic tool on top of it. Eclipse provides an open platform for both GUI and non-GUI based application development tools to run on a wide range of operating systems. Eclipse thus has an open extensible structure based on plug-ins, a plug-in being the smallest unit of functionality that Eclipse recognizes. A plug-in provides mechanisms for extending itself by defining extension points. Other plug-ins can extend these extension points by defining extensions.

5. OUR APPROACH

We propose a new approach to workflow modeling and analysis. Workflow modeling needs a language that is intuitive and easy to use. At the same time, mere modeling of workflow is not sufficient. It is necessary to analyze a workflow model to look for possible flaws and further improvements. As for the modeling language Activity diagrams from UML 2.0 provide a good option. For analysis we need a more formal tool, and we get that in Petri nets. So we map the workflow model into corresponding Petri nets model for analysis. Finally we implement this on Eclipse. Above mentioned aspects are discussed in details below:

5.1 ACTIVITY DIAGRAMS for MODELING WORKFLOWS

Activity diagram from UML 2.0 provides all the basic constructs needed. Major constructs for workflow modeling are sequence, parallel path, alternative path and iteration. Activity diagram constructs, *start, end, fork, decision*, and *activity* can be used for modeling all these constructs. *Start* can be used to indicate beginning of a process where as *end* can be used to indicate the end of a process. *Fork* can be used for splitting a process into several parallel execution paths. *Decision* can be used for providing alternative paths. We can also model iteration by connecting two decisions.

5.2 ACTIVITY DIAGRAM to PETRI NETS

An Activity diagram can be mapped to a Petri net, which includes all kinds of control flow [4]. Here activity and fork nodes are mapped to Petri net transitions and start, end, and decision nodes are mapped to places. Connections are mapped in such a way that always there is an arc either from transition to place or place to transition.

Figure 1 shows an Activity diagram and Figure 2 shows the corresponding Petri net obtained after converting it.

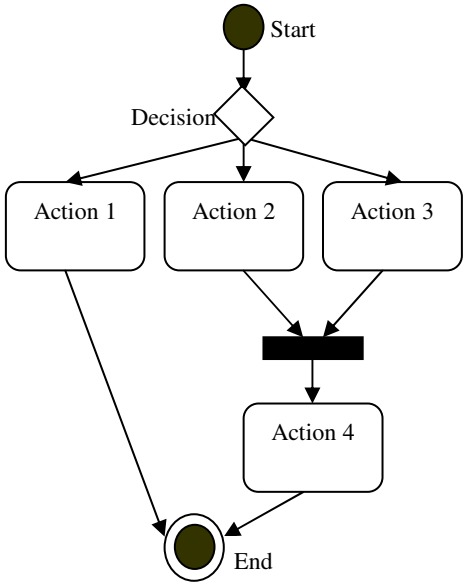

Figure 1. An example of Activity diagram

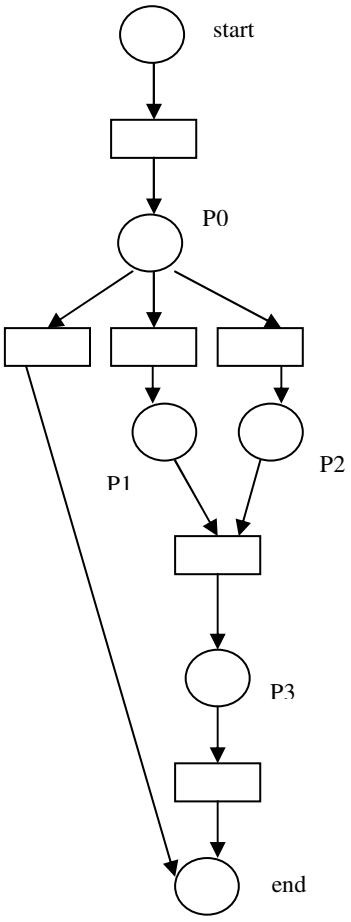

Figure 2. Petri Net mapping of Fig. 1

5.3 PETRI NETS for ANALYZING WORKFLOWS

A huge amount of work has been done on Petri nets so far and hence a large number of results are available. One needs to find out a set of results, which can help in analyzing workflows. Two such useful methods are coverability tree and incidence matrix.

A coverability tree is actually a reachability tree with some modification to take care of the case when the given Petri net is not bounded.

The coverability tree of a given Petri net with initial marking M0 is constructed using the following algorithm [2]:

1. Label the initial marking M, as the root and tag it "new."
2. While "new" markings exist, do the following:
    a. Select a new marking M.

If M is identical to a marking on the path from the root to M, then tag M

1. "old" and go to another new marking.
2. If no transitions are enabled at M, tag M "dead-end."
3. While there exist enabled transitions at M, do the following for each enabled transition t at M:
    a. Obtain the marking M' that results from firing t at M.
    b. On the path from the root to M if there exists a marking M" such that M'(p)>M"(p) for each place p and M'≠M", i.e., M" is coverable, then replace M'(p) by ω for each p such that M'(p) > M"(p).
    c. Introduce M' as a node, draw an arc with label t from M to M', and tag M' "new."

We can also use an Incidence matrix [2] to calculate reachable marking from a given marking after firing a particular transition.

Using both coverability tree and incidence matrix we can study some properties of Petri nets, which are helpful in analyzing corresponding workflow from which the Petri net has been constructed. Three such useful properties are boundedness, safeness, and deadlock. If we start with an initial marking where there is only one token in the start place and no token in other places, then absence of boundedness indicates that a particular place have infinite number of tokens. So this indicates that we can never reach the end place without having left some tokens in other places. In workflow domains this implies that we can never end an activity without leaving some reference to it.

Safeness property in workflow domain will ensure that we don't have more than reference to an object to be processed. This makes sense since there is no need of processing two same objects when one is needed.

Deadlock property is very useful from workflow point of view as it indicates that the corresponding workflow has some activity which cannot be reached hence the design has some flaws.

From the above discussion it is clear that from Petri net analysis we can often comment on the properties of a workflow.

5.4 IMPLEMENTING as ECLIPSE PLUG-IN

We demonstrate the possibility of our approach by developing an Eclipse plug-in [19]. The flow of our tool on Eclipse is as given in Figure 3. A workflow editor is provided which can be used for modeling workflow using activity diagrams syntax. Then user can convert the workflow model to corresponding Petri net model. We generate the Petri nets in PNML format [3], which is a standardized XML based interchange format for representing Petri Nets. This is useful for importing and exporting a Petri net model. This Petri net model, in turn, may be fed into the Petri net analyzer, which does above-mentioned analysis using which we can comment on the original workflow model.

We implement the activity diagram editor using a well-known design pattern called the model-view-controller (MVC) [18]. As the name suggests, MVC consists of three kinds of objects. The model represents the actual thing that is persisted and restored. The view represents how we display the model. Controller handles user input. In MVC pattern, model and view remains decoupled from each other. This helps in increasing flexibility and reuse. Also we can have multiple views of the same model. Using already available features of Eclipse, we could implement MVC. For implementing the view part we use SWT and Draw2d frameworks of Eclipse. For implementing the controller part we use the Graphical Editing Framework (GEF) of Eclipse.

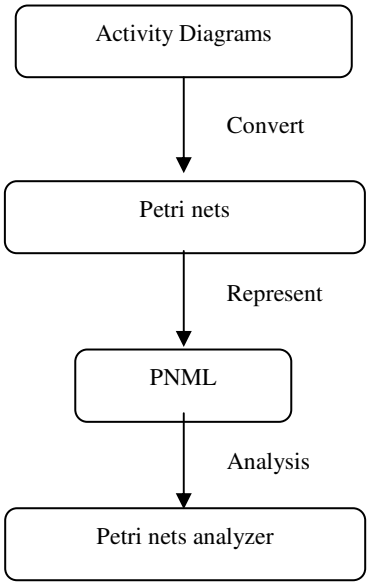

Figure 3. Dataflow of our tool

6. SCREENSHOTS OF OUR TOOL

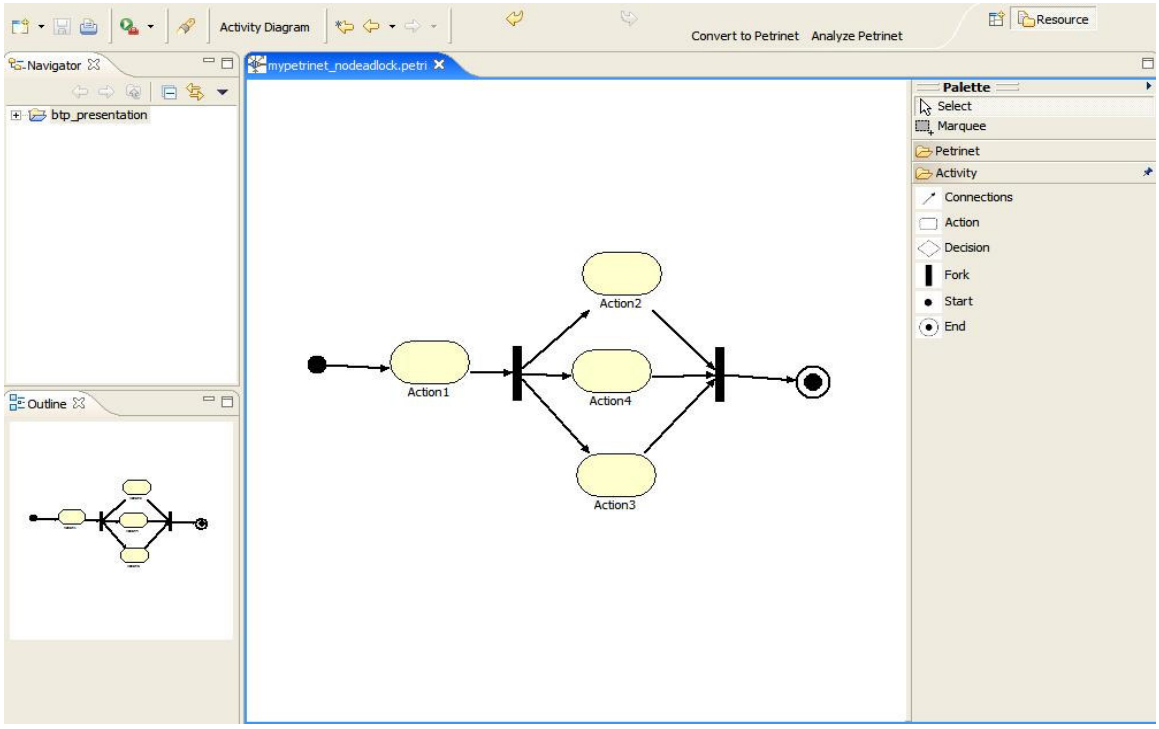

Figure 4. Screen shot showing an Activity diagram

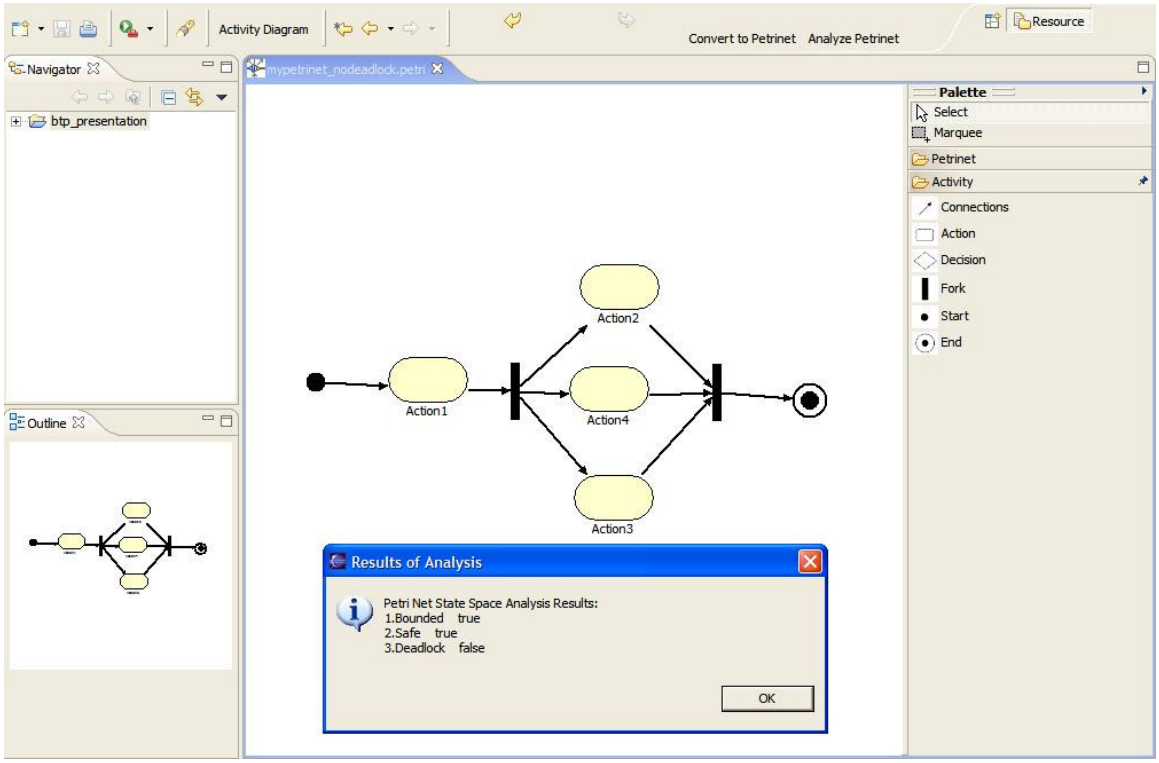

Figure 5. Screen shot showing the results of analysis

7. RELATED WORK

Van Der Aalst et al has proposed Petri nets for both modeling and analyzing workflows in [11], [12], [13], [14]. Petri nets have been enhanced with time, color and hierarch to enhance their modeling power [15]. The problem with this is, Petri nets are not an easy language for modeling workflows. However, there are not many theoretical results available with high-level Petri nets, which can be used for analysis[2].

Activity diagram has been argued by many as an alternative for modeling workflows. After Van Der Aalst et al identified workflow patterns [9], it has been shown that they can be modeled using Activity diagrams [10]. There have been efforts for defining semantics for activity diagram, so that execution of the workflow models can be done ([6], [7], [8]).

8. CONCLUSION AND FURTHER WORK

We have proposed a new way of looking at analysis of workflows. Modeling of workflows should be done in a language, which is easy and more intuitive to work with like Activity diagrams. But analysis has to be done in a more formal language like Petri nets. We demonstrate this by giving an Eclipse based tool, which can model workflows using Activity diagrams, and then analyze the model using Petri nets. We have so far mentioned three properties of Petri nets, which are useful in commenting on workflow models. More such properties can be looked for and at the same time more constructs from activity diagrams can be added.

9. REFERENCES

[1] W.M.P van der Aalst and Kees van Hee, *Workflow Management, Models, Methods, and Systems*

[2] Tadao Murata, *Petri Nets: Properties, Analysis and Applications,* Proceedings of the IEEE, Vol. 77, No. 4

[3] Billington et al., *The Petri Net Markup Language: Concepts,Technology, and Tools* [Online]. Available: http://www.informatik.huberlin.de/top/pnml/download/about/PNML_CTT.pdf

[4] Harald Storrle. "Semantics of Control-Flow in UML 2.0 Activities," *vlhcc*, pp. 235-242, 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC'04), 2004.

[5] Workflow management coalition [Online]. http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[6] Rik Eshuis, Roel Wieringa. *A formal semantics for UML Activity Diagrams – Formalising workflow models,* Technical Report CTIT-01-04, U. Twente, Dept. Of Computer Science, 2001.

[7] Rik Eshuis, Roel Wieringa. *Verification support for workflow design with UML activity graphs*, In Proc.24th Intl. Conf. on Software Engineering (ICSE'02), pages 166-176. IEEE, 2002.

[8] Rik Eshuis, Roel Wieringa. *A real time execution semantics for UML activity diagrams,* In H. Hussmann, editor, Proc. 4th Intl. Conf. Fundamental approaches to software engineering (FASE'01), number 2029 in LNCS, pages 76-90. Springer Verlag, 2001.

[9] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Workflow Patterns*, BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000

[10] Stephen A white, *Process Modelling Notations and Workflow patterns.* [Online] http://www.omg.org/bp-corner/bp-files/Process_Modeling_Notations.pdf

[11] W.M.P. van der Aalst. *The application of Petri nets to workflow management*, The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[12] W.M.P. van der Aalst. Woflan: *A Petri-net-based Workflow Analyzer*, Systems Analysis - Modelling - Simulation, 35(3):345-357, 1999.

[13] W.M.P. van der Aalst. *Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques*, In Business Process Management: Models, Techniques, and Empirical Studies, *volume 1806 of* Lecture Notes in Computer Science, pages 161-183. Springer-Verlag, Berlin, 2000.

[14] W.M.P. van der Aalst and A.H.M. ter Hofstede. *Verification of Workflow Task Structures: A Petri-net-based Approach*, Information Systems, 25(1):43-69, 2000.

[15] W.M.P. van der Aalst, K.M. van Hee, G.J. Houben, *Petri nets and related formalisms*, Proceedings of the second Workshop on Computer-Supported Cooperative Work.

[16] Shavor et al., *The Java Developer's Guide to Eclipse*, 3rd ed, Addison-Wesley, 2003.

[17] [Online] http://www.eclipse.org

[18] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns,* Addison-Wesley.

[19] http://www.cse.iitk.ac.in/~soft_arch/petri/