A Study of Information Systems model for Activity diagram Analysis

CHIEN-YUAN LAI¹, DONG-HER SHIH², HSIU-SEN CHIANG³, CHING-CHIANG CHEN⁴*

¹, ², ⁴Department of Information Management
National Yunlin University of Science and Technology
³Department of Information Management
Da-Yeh University, Taiwan
No.123, Sec. 3, Dasyue Road., Douliou City, Yulin Country, 640
Taiwan, R.O.C.

g9723803@yuntech.edu.tw; shihdh@yuntech.edu.tw; chianghs@mail.dyu.edu.tw; g9723802@yuntech.edu.tw.*

Abstract:- Unified Modeling Language (UML) is the Object Management Group (OMG) standard notation for object-oriented modeling. UML activity diagram (AD) are widely used to workflow and system flow in system analysis. However, activity diagram lacks support for simulation, dynamic semantics limits and verifiability capabilities. APN are a popular technique for modeling the control flow dimension of workflows and APN has a complete semantics, simulation and verifiability capabilities. Therefore, the main purpose of this study is using APN to improving activity diagram of drawback and we propose an approach to transform activity diagram into APN. Finally, a case study taken from a dynamic travel recommended system case is used for explaining and illustrating this concept.

Key-Words: - UML 、Associative Petri Net 、activity diagram 、Transform algorithm

1 Introduction

Unified Modeling Language (UML) is the Object Management Group (OMG) standard notation for object-oriented modeling. It is easy, graphical and appealing, but in several cases still too imprecise. UML is considerably easy to use but it does not support formal model analysis because it does not have a formal semantics. UML is strong as modeling means, supplies several different diagrammatic notations for representing the different aspects of a system under development, but lacks simulation and verifiability capabilities. UML 2.0 is composed of several diagram types (activity, sequence, use case, class, timing and many others). activity diagrams have been added to the UML rather late. They have always been poorly integrated, lacked expressiveness, and did not have an adequate semantics. UML 2.0 replaces UML 1.5 ‘activity diagram’ concepts based on state machines with activity modeling that is supposedly based on Petri Net semantics (Borger et al., 2000). Activities are suitable to model web processing, web service composition (Artagna & Pernici, 2007), business process modeling, workflow management systems, system integration and even basic software operation. The following are some main properties of UML 2 activity diagrams.

Activity diagram nodes have flow-of-control constructs like synchronization, decision, concurrency and sequence. These are fundamentally similar to those of Associative Petri Nets. Activity diagram semantics are based on token flows. Tokens can contain objects, data, control information. Tokens are normally distinguishable through an individual time-stamp. UML 2 activities try to deal with many different levels of activities: 1) fundamental 2) basic 3) Intermediate 4) complete 5) structured, 6) complete structured and 7) extra structured. Each level adds its own constructs addressing a particular area. E.g. structured activities address traditional programming language modeling. Other activity classes like fundamental and basic are ideal for high level modeling and business process modeling. These classes include task sequencing, data flows and control flows based on normal resources. The most suited to convert into Associative Petri Net models are fundamental, basic and intermediate activities.

Associative Petri Nets (Shih, et al., 2007) are a popular technique for modeling the control flow dimension of workflows. When modeling workflows, people tend to draw nodes that represent tasks or activities, and arrows between the nodes that represent sequencing of activities. The resulting
diagrams look like Associative Petri Nets, and so Associative Petri Nets seem a natural technique for modeling workflows (Ellis, & Nutt, 1993). The following arguments are often used to support this: Associative Petri Nets are graphical, they have a formal semantics, they can express most of the desirable routing constructs, there is an abundance of analysis techniques for proving properties about them, and finally they are vendor-independent. Some of reasons following items are UML transform into Associative Petri Net (Petriu & Shen, 2002; Kristensen, et al., 2004; Baresi & Pezze, 2007). 1)UML lacks support for strong simulation and analysis techniques. 2)APN is graphical formalisms. 3)APN model all UML activity constructs. 4)APN can be verified and simulated. 5)APN has mathematical properties. 6)APN is suitable for visualization In this paper, we propose an approach of activity diagram transform into Associative Petri Nets notation and improving the some of drawback. This work can improve of the simulation and verifiability capabilities for activity diagram. A case study taken from a dynamic travel recommended system case is used for explaining and illustrating this concept.

2 Literature Review

The related work is structured as follows: section 2.1 we will introduction for activity diagram and activity subnet. Section 2.2 we will introduction Associative Petri Net. Section 2.3 is mapping activity diagram to associative Petri Net.

2.1 Overview of the activity diagram

Unified Modeling Language (UML) is the Object Management Group (OMG) standard notation for object-oriented modeling. It is easy, graphical and appealing, but in several cases still too imprecise. Activity diagrams are a technique to describe procedural logic, business process, and work flow. In many ways, they play a role similar to flowcharts, but the principal difference between them and flowchart notation is that they support parallel behavior (Fowler, 2004).

An activity diagram is a special form of state machine intended to model computations and workflows (Fowler, 2004). The states of the activity diagram represent the states of executing the computations, not the states of an ordinary object. Normally, an activity diagram assumes that computations proceed without external event-based interruptions.

An activity diagram contains activity states. An activity state represents the execution of a statement in a procedure of the performance of an activity in a workflow. Instead of waiting for an event, as in a normal wait state, an activity state waits for the completion of its computation. When the activity completes, then execution proceeds to the next activity state within the graph. A completion transition in an activity diagram fires when the preceding activity is complete. Activity states usually do not have transitions with explicit events, but they may be aborted by transitions on enclosing states. (Rumbaugh, et al., 1999)

An activity diagram may also contain action states, which are similar to activity states, except that they are atomic and do not permit transitions while they are active. Action states should usually be used for short bookkeeping operations.

An activity diagram may contain branches, as well asorking of control into concurrent threads. Concurrent threads represent activities that can be performed concurrently by different objects or persons in an organization. Frequently concurrency arises from aggregation, in which each object has its own concurrent thread. Concurrent activities can be performed simultaneously or in any order. An activity diagram is like traditional flow chart except it permits concurrent control in addition to sequential control-a big difference. (Rumbaugh, et al., 1999)

The activity diagram contains seven type of state: 1) Action state, 2) Initial/Final action state, 3) Transition, 4) Branching, 5) Fork and join, 6) Object flow, 7) Swimlanes. Take the seventh item as the example, Swimlanes:

The activities in an activity diagram can be partitioned into regions, which are called swimlanes from their visual appearance as regions on a diagram separated by dashed lines. A swimlane is an organizational unit for the contents of an activity diagram. It has no inherent semantics, but can be used as the modeler desires. Often, each swimlane represents an organizational unit within a real-world organization.

It is often useful to organize the activities in a model according to responsibility. For example, by grouping together all the activities handled by one
business organization. This kind of assignment can be shown by organizing the activities into distinct regions separated by lines in the diagram. Because of their appearance, each region is called a swimlane. For example, see Fig. 1.

![Swimlanes and Object Flow](image)

**Fig. 1 Swimlanes and Object Flow**

### 2.2 Associative Petri Nets (APN)

Petri Nets (PN) has its origin in Carl Adam Petri’s dissertation, submitted in 1962 to the faculty of Mathematics and Physics at the Technical University of Darmstadt, West Germany. PN is a graphical and mathematical modeling tool applicable to many systems. (Murata, 1989) They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. However, APN differs from current modeling tools, as it is belong to dynamic modeling, except provide with parallel and distributed system of dynamic behavior, on other hand they support of graphical for simple hierarchical theory and support mathematics for qualitative and quantitative analysis, further development on special flow.

Associative Petri Nets (APN) has its origin in Dong-Her SHIH et al. in 2007. The concept of APN is derived from FPN (Chen, et al., 1990). We shall derive the definition of a generalized APN and augment the association production rules (APRs) of APN by adding association rule operators and equipping the nets with reasoning facilities. Based on the generalized associative Petri Net model, a systematic procedure of Associative Petri Net model construction methodology had been proposed.

### 2.3 Definition of APN

An APN is a directed graph, which contains three types of nodes: places, squares, and transitions where circles represent places, squares represent thresholds of association degree, and bars represent transitions. Each place may contain a token associated with a truth-value between zero and one. Each transition is associated with a CF value between zero and one. Directed arcs represent the relationships between places. A generalized APN structure, as shown in Fig. 2, can be defined as a 13-tuple:

\[ APN = (P, T, S, D, \{I, O, C, a, b, W, Th\}) \]

where
- \( P = \{p_1, p_2, \ldots, p_n\} \) is a finite set of places,
- \( T = \{t_1, t_2, \ldots, t_n\} \) is a finite set of transitions,
- \( S = \{s_1, s_2, \ldots, s_m\} \) is a finite set of supports,
- \( D = \{d_1, d_2, \ldots, d_n\} \) is a finite set of propositions,
- \( \{r_1, r_2, \ldots, r_m\} \) is a finite set of thresholds of the supports,
- \( \{r_1, r_2, \ldots, r_m\} \) is a finite set of thresholds of the confidences,
- \( P \rightarrow T \rightarrow D = f, |P| = |D| \),
- \( I : T \otimes P^X \) is an input function, a mapping from transitions to bags of places,
- \( C : T \otimes [0,1] \) is the confidence degree of relationship between places,
- \( a : P \otimes [0,1] \) is an association function, a mapping from places to real values between zero and one,
- \( b : P \otimes D \) is an association function that assigns a real value between zero to one to each support, and
- \( Th : S \rightarrow Th \) is an association function that defines a mapping from support to thresholds.
2.4 Mapping activity diagram to APN

Activity diagrams are a technique to describe procedural logic, business process, and workflow. It is a graphical and appealing. However, in many ways activity diagram lacks support for simulation, formal semantics and verifiability capabilities. APN is a graphical and strong modeling tool. In many ways, they play a role similar to activity diagram, and APN has formal semantics, simulation and verifiability capabilities. Therefore, in this paper we propose APN to improving activity diagram for system analysis and workflow. Activities and corresponding Associative Petri Net notation activity diagram are lack of formal semantic, simulation and verifiability capabilities, therefore we propose an APN building through activities. The various characteristics of activities are as follows: Action node to conversion: Action node is a rounded rectangular, in rectangle marked with its activities, status and process. It is can express action state and process, the theory is similar to Place by APN. The Place is a cycle, its express activity flow and state by token transition, and marked its activity properties on a nearby Place. The Fig.3 is show action node mapping to Place.

A methodology based on a five-stage schema was developed for activity diagram transfer to Associative Petri Net. The five stages correspond to 1) Initial node determination, 2) Finding basic structure models, 3) Final node determination, 4) AD-to APN model mapping, 5) Models combination, 6) APN model accomplishment.
3.1 Initial node determination

In this step, we shall incise the starting place \( P_i \). If a node is no parent node then called this node is a start node \( P_i \), we determine a start node. Next step, we decide an incision place for model combine, therefore, we will incision next node of \( P_i \) to facilitate follow-up action. Finally, we will map \( P_i \) and next node of \( P_i \).

3.2 Finding basic structure models

According to section 2.3 basic structure models contains five types: 1) activity sequence structure, 2) activity fork and join structure, 3) activity branching structure, 4) activity iteration structure, 5) activity object flow structure. On the other hand, we facilitate for model combine, therefore, we shall incision the basic model for parent node and child node.

3.3 Final node determination

In this step, the main purpose of decide the final node of activity diagram, and determination the model range, and mapping the Final node. Therefore, we decide a final node, if a node is no child node then called this node is a final node \( P_f \). Through the Pf, we can judge the action of activity diagram between \( P_i \) and \( P_f \). Due to \( P_f \) is no child node, therefore, we will incision \( P_f \) and present node of \( P_f \) for combine point. Finally, we will map \( P_f \) and present node of \( P_i \).

3.4 AD-to-APN model mapping

As described in Section2.3, the activity diagram is transformed into an APN.

3.5 Models Combination

According to the above steps to combined. Owing to activity diagram is a directed graph, and workflow transmitted by Transition. For that reason, it’s the same as the required permutations and combinations, otherwise this graph through the conversion would be a error graph. Therefore, we shall for each Place state to match, if the place of the same state, then combine this model and just leave one place, and repeat above then can output a goal graph.

4 Result

Information Systems model for Activity diagram Analysis is very important, there is no research for uml activity diagram to study the related lack of improvement, this paper proposes to \( p_\text{net} \) ways to solve the problem in this paper, The handling of these shortcomings, for uml activity diagram related to the improvement and use, will receive great benefits. According to above steps, we finished the APN model and obtain a structure of an APN. A detailed construction algorithm of AD-to-APN is shown in Fig. 5.

Fig. 5 Construction algorithm of an AD to APN model


5 Conclusion

The activity diagrams are widely used to workflow and system flow in system analysis. However, activity diagram lacks support for simulation, dynamic semantics limits and verifiability capabilities. Therefore, in this paper, we propose a transform methodology with AD-to-APN model. This work can improve of the simulation and verifiability capabilities for activity diagram. Finally, a case study taken from a dynamic travel recommended system case is used for explaining and illustrating this concept.

References: