ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*

# Verification of Interaction Overview Diagrams Using Colored Petrinets

Vinai G. Biju [1], Vinod. K. Agrawal[2]

Assistant Prof. Dept. of CSE, Christ University, Bangalore, India[1]

Professor, Dept. of Information Science, PESIT, Bangalore, India[2]

**Abstract**: The UML 2 Interaction Overview Diagram (IOD) provides a visual representation of system's overall interactions. The UML 2 IOD visualizes the behaviour of a system only for the interaction among the components and also a partial order between send and receive message events; however the semantics of communication among the interaction occurrences and process execution policy remains vague. It does not provide formal approach of specification and has a weak support for the validation. So, it is extremely important to improve the 'quality' of the design model using formal description and thereby validating the highest level of abstraction of design. An attempt has been made in this paper towards formalizing the IOD Sequence diagrams by mapping it into Colored Petri nets (CPNs). This approach of formal translation allows a designer using UML 2.0 to verify and validate models using CPN tools.

**Keywords**: Colored Petri net (CPN), Unified Modeling Language (UML), and Interaction Overview Diagram (IOD).

## I. INTRODUCTION

This Interaction Overview Diagram (IOD) combines the power of sequence diagram and activity diagram together. It can be used to describe an overview of a complex system by embedding the objects of Activity Diagram, Inline interaction or Interaction Occurrences inside a control flow structure. IOD provides high level structuring mechanism for sequence diagrams [1]. Even though UML 2 brings more precision than UML 1.x, it remains informal and lacks tools for automatic analysis and validation [1]. Compared to UML 1.x, the concrete syntax of activity diagram has remained mostly the same, but the abstract syntax and semantics have changed drastically.

IOD illustrates dependence between the important sequences of a system, which can be presented by an activity diagram. The notations used in IOD incorporate constructs from sequence diagrams with fork, join, and decision and merge nodes from activity diagrams. While in UML 1.x, activity diagrams have been defined as a kind of state machine diagrams, but in UML 2 there exist no relation between the two diagrams and the meaning of activity diagrams is being explained in terms of Petri net notions like token, flow, edge-weight and so on. In the same way, sequence diagrams have been extended considerably, and they have approximately the same expressive power as High Level Message Sequence
Charts (MSCs) [2]. IODs are special kinds of activity diagrams where the activity nodes are actions or interactions and the activity edges denote the control flow.

A good deal of research has already been dealt with the semantics of UML 2 activity and sequence diagrams [4], [5], [6], [7] but only few results are communicated on the formalization of the IODs which may be used to combine interactions into a kind of dataflow resonant of activity diagrams, where the places of activity-states are taken by interactions.

The main motivation for this paper is to create a constructive approach to derive a CPN model which realizes the same scenario as that of IOD. Thus a formal representation representing multiple scenarios composing an interaction overview diagram is designed using CPN that has a unique interpretation and allows the analysis and synthesis of implementation [15], [16], [17], [22]. If the same scenario occurs twice then the second instance of the scenario starts only after the first instance of scenario has been completed resulting in safe models i.e. all the events of scenario have been occurred satisfactorily.

**Literature review:**
Störrle [3] analyzed the UML 2 activity diagrams semantics and proposed an approach to their formalization. Staines [4] proposed a suitable formalism to achieve the transformation from UML 2 activity diagrams to Petri nets. Lam [5] formalized the execution semantics of activity diagram using the π-Calculus. This formalization provides a theoretical foundation as well as a starting point for building automated software tool. Cengarle and Knapp [6] have provided an operational semantics to UML 2 interactions. They

furthermore have addressed the lack of UML interactions to describe explicitly the variability and proposed extensions equipped with denotation semantics [7]. Knapp and Wuttke [8] translated the UML 2 interactions into automata and then verified that the proposed design meets the requirements stated in the scenarios by examining models. Kloul and Küster-Filipe [9] illustrated how to model mobility using IODs and proposed a formal semantics to the latter by translating them to the stochastic process algebra PEPA nets.

**Paper Outline**: Section 2. gives a background of colored petri nets. Methodology of transformation from IOD to CPN is detailed in section 3. Verification of the CPN model is discussed in section 4. A case study has been included in section 5. Results after verification and analysis using CPN model are described in section 6 and section 7. State spaces and report are generated in section 8. Conclusions & future scope are mentioned in section 9

## II. Background: Function of Colored Petri Net (CPN) models

The UML 2 IOD visualizes the behavior of a system only for the interaction among the components and also a partial order between send and receive message events; however the semantics of communication among the interaction occurrences and process execution policy remains vague. It does not provide formal approach of specification and has a weak support for the validation. So it is extremely important to improve the 'quality' of the design model using formal description and thereby validating the highest level of abstraction of design. CPN tools can be used to verify properties like Home, Liveness, and Fairness for any complex system [18]. It is assumed that the reader has prior knowledge of properties of CPN. The CPN model is synthesizable using well established software and hardware synthesis techniques [14].

CPN model consists of data, places, transitions and arcs. Location for holding data is known as a place and the actions are represented by transitions. Places and transitions are connected by a directed arc which specifies the data flow paths. The places in CPN model are named from the preconditions or objects passing the messages. CPN color sets and variables are defined in the global area of the CPN model. Tokens represent the data objects and the Color set defines the token type. Tokens of a particular color are placed in locations called places.

CPN consists of:

| | |
|---|---|
| P: | set of places. |
| T: | set of transitions. |
| A: | set of arcs. |
| Σ: | set of colour sets. |
| V: | set of variables. |
| C: | colour set function (assigns colour sets to places). |
| G: | guard function (assigns guards to transitions). |
| E: | arc expression function (assigns arc expressions to arcs). |
| I: | initialisation function (assigns initial markings to places) |

A guard function G: T $\rightarrow$ expression assigns a guard to each transition. The Guard expression is evaluated to a Boolean value. $[G(t)] = \text{Bool}$for all $t \in T$

A set of directed arcs A can be represented by $A \subseteq P \times T \cup T \times P$. Each arc starts in a place and ends in a transition or it start in a transition and ends in a place.

An arc expression function E: A$\rightarrow$ expression assigns an arc expression to each arc.

$[E(a)] = C(p)_{MS}$for all $a \in A$, where $p$ is the place connected to the arc $a$. Arc expression evaluates to a multiset of tokens belonging to the colour set of the connected place denoted by $C(p)_{MS.}$

A marking is a function M mapping each place $p$ into a multiset of tokens $M(p) \in C(p)_{MS}$. All token values must belong to the colour set of the placeB(t). A binding element is a pair *(t,b)* such that *t* is a transition and $b \in B(t)$.The set of all binding elements of a transition t is denoted BE(t) [10], [11].

## III. METHODOLOGY FOR TRANSFORMING IOD TO CPN

A mapping rule to transform an interaction overview model of a system as in Fig.1 to CPN model is discussed in this section. Initially each fragment of interaction occurrences is converted to intermediate nets. Then these intermediate nets are combined to form a single net by

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*

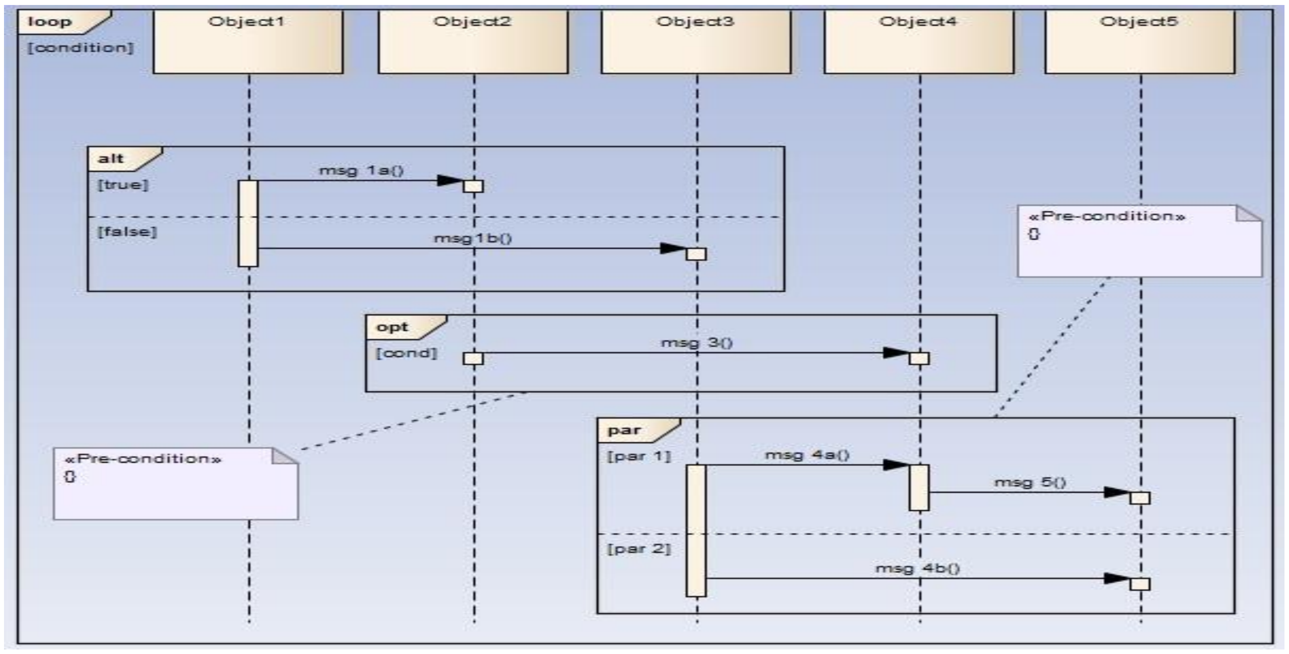


**Fig. 1: Loop, Alt , Opt and Par -sequence Fragments of IOD.**

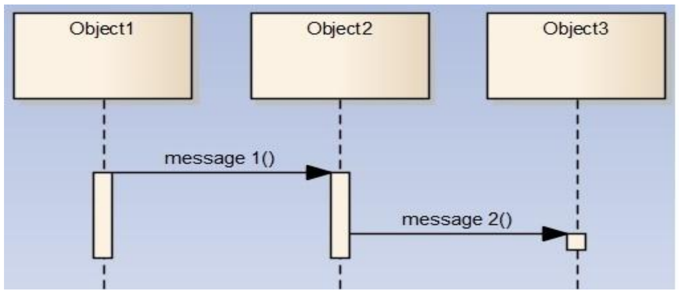
merging the transitions corresponding to the messages of sequence and communication diagrams. Messages passed between the objects in communication diagram are also mapped to transitions in CPN. The intermediate net in Fig. 2b with transitions having label *message1* connected from places *object1/precondition1* and *object2 /precondition2* is the same message instance of the sequence diagram sent from *object 1 to object 2.*

**Algorithm:** Mapping rule for IOD Sequence Fragments

```
Input:
    IOD Sequence Fragments $f_i$
Output:
A Coloured Petri Net model
m = (P, T, A, Σ, V, C, G, E, I)
// assign transition $t_i$ in $m$ for each $f_i$
for each Sequence fragments from top to bottom in the life line do
    $f_i \Rightarrow t_i,\ m.T = m.T \cup \{t_i\};$
end for
//assign place $p_i$ to each $t_i$ in $m$
for each Transition $t_i$ of $T$ in $m$ do
    $m.P = m.P \cup \{p_i\};$
end for
//Add arcs:
for each transition $t_i$ in $m$ do
    $A \subseteq P \times T \cup T \times P$
//Add Arc Expression $E(a)$ from colour set $C(p)_{MS}$ and variables of arc Expression $Var(E(a))$
    $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \sum$
//Arc Expressions $E(a)$ from each node $x_i$ and $x_k$ is mapped from the messages $m_i$ passed between objects $Ob_i$ and $Ob_k$ of IOD sequence fragment $f_i$
    $\forall (x_1, x_2) \in (P \times T \cup T \times P) : [E(x_i, x_k) = \sum_{a_i \in A(x_i, x_k)} E(a)] = \sum_{m_i \in f_i(Ob_i, Ob_k)} M(a)$
//Assign Guard Functions $G(t)$ with respect to Condition and Guards of loop, alt and opt sequence fragments
Guard conditions in sequence fragments $\Rightarrow G(t)$,
    $m.G = m.G \cup \{G(t)\}$
    $\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \sum$
        $\forall t \in T : [Var(t) = \{v \mid v \in Var(G(t)) \vee$
        $\exists a \in A(t) : v \in Var(E(a))\}]$
end for
```

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*



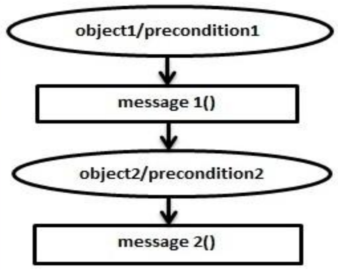


**Fig. 2 (a) Successive messages in sequence diagram** **Fig. 2(b) CPN model**

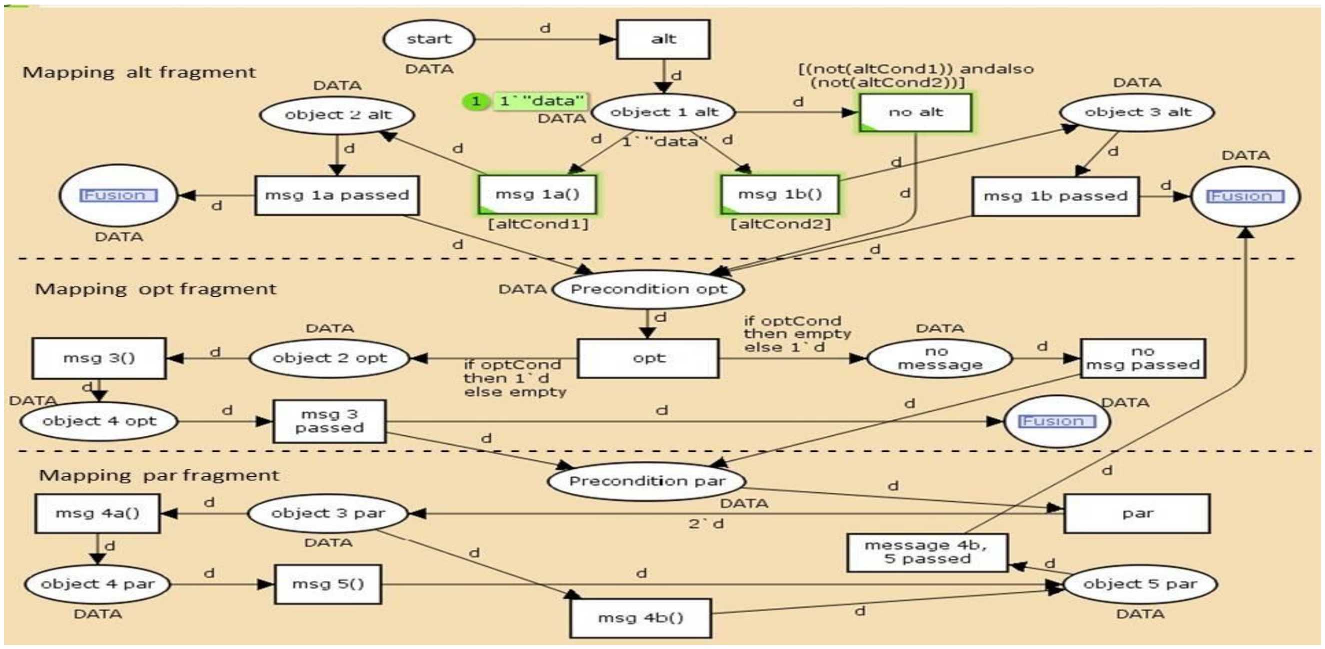


**Fig. 3: Transforming alt, opt, par block to CPN model**

A message passed within the alt fragment between objects is mapped to a transition in CPN model. Here from Fig.1 and Fig.3, *msg1a()* that is passed from *Object 1 to Object2* is mapped to the transition *msg 1a()* with *[altCond1]*guard function. If the *[altCond1]* guard evaluates to false then, *msg1b()* that is mapped to a transition in Fig.3 is fired. The guard *[altCond2]* can be used for expressions that represents 'false' in the alt fragment. The system is also modeled if *'alt'* fragments have no expressions evaluated to true. This is represented by the transition *'no alt'*. Soon after *'alt'* fragment message is passed on, we receive tokens on aplace named *'Precondition opt'* as shown in Fig.3 to indicate that the system can start with next message to be passed along the next sequence fragment. *'Opt'* Fragment in IOD sequence fragment is mapped to the transition named *'opt'* in Fig.3. The Boolean expression *'cond'* of opt fragment in Fig.1 is mapped to arc expressions in CPN model as '*If OptCond then 1'd else empty*' in Fig.3. The arc expression *1'd* denotes the message passing of 1 data token as we have only 1 message passed between objects. Each object like *Object1*, *Object2* etc. of Fig.1 is mapped to corresponding '*place*' in Fig.3. The messages *msg 4a()*and *msg 4b()* send in parallel from Object 3 to Object 4 and Object 3 to Object 5 in the parallel construct sequence fragment (*par*) are represented in CPN as follows: The '*par*' sequence fragment of Fig.1 is mapped to a transition named '*par*' in Fig.3. Arcs are included evolving out of the transition *'par'* to symbolize parallel firing of tokens. A *'fusion'*place is added to take a note of the total successful transfer of messages of individual sequence fragments.

## IV. VERIFICATION AND ANALYSIS OF CPN MODEL

Once the model has been compiled, the behavior of the system can be investigated by means of simulations using CPN tools. These simulations typically have the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*

characteristics of single step debugging in which the movement of token is observed in great detail, and the user chooses the next binding elements to occur. During which, the markings of the places are shown directly on the CPN diagram similar to Fig.3. It typically reveals some shortcomings and errors in the CPN model which then have to be resolved. Hence, the first phase normally consists of a number of iterations switching back and forth between the editor and the simulator, then, gradually refining and improving the CPN model. The simulation/execution of the CPN model is driven by the simulator engine of CPN ML.

Concurrent scenarios where multiple objects communicate within an interaction fragment of an IOD can be visually represented and executed in CPN thereby dynamic visualization and changes can be modeled and viewed. Scenarios where objects of communication and sequence diagrams that never communicate can be found out by executing the model in CPN Tools checking Liveness Properties. Dependent objects within IOD can be identified in detail and checked for loops using state space generated from the CPN model. Conditions for the objects to communicate within IOD can be checked and verified using CPNML language. Finally the state of the systems design represented by IOD after 100 or more iterations of concurrent message passing between Interaction fragments can be monitored using the IOD transformed CPN model. The transition will only be enabled only when the binding element *b = <x=id, d=data>* is evaluated to true. Transitions in addition to expressions has a “guard” [variable = desired object] which is a Boolean condition. Adding guard makes the design more robust towards errors. The *predecessor* and *successor* for each object passing the messages can be plotted with state space graphs using *StateSpace* tool. The sequence of message passing between objects can be checked for infinite loops by fairness properties of the report generated with state space graphs and Strongly connected components. Monitors can be included in the design like “breakpoints” to stop the simulation and identify the scenario once a particular condition is fulfilled or an object receives a message in IOD.

## V. A CASE STUDY

In order to demonstrate the practical usability of the proposed mapping process, the requirements for a weblog Content Management System (CMS) is taken into consideration as a case study. Weblogs are commonly referred as blogs, originally started out as privately maintained web pages for authors to write about anything, such as personal details, job postings, marketing of products etc. These days, blogs are usually packaged into an overall CMS. The Administrator interacts with the system to create a new blogger's account.

The content management system shall allow an administrator to create a new blog account, provided the personal details of the new blogger are verified using the author credentials database. Bloggers submit new entries to the system, and the administrator allocates new blog accounts. A well-publicized blog can attract thousands of readers. A complete use case description for the "Create a new Blog Account” may be described as: a new or existing author requests a new blog account from the Administrator. The system is limited to recognized authors and so the author needs to have appropriate proof of identity. A successful end condition generates a new blog account created for the author. The sequence of steps can also be detailed as:

i. Administrator asks the system to create a new blog account
ii. Administrator selects an account type.
iii. Administrator enters the author's details.
iv. Author's details are verified using the Author Credentials Database.
v. New blog account is created.
vi. A summary of the new blog account's detailsare emailed to the author.

If the steps required to create each of these accounts in CMS differ slightly from the original use case, then it is required to describe the general behavior for creating a blog account captured in the corresponding use cases. Later specialized use cases are defined in which the account being created is a specific type, such as a regular account with one blog or an editorial account that can make changes to entries in a set of blogs.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*

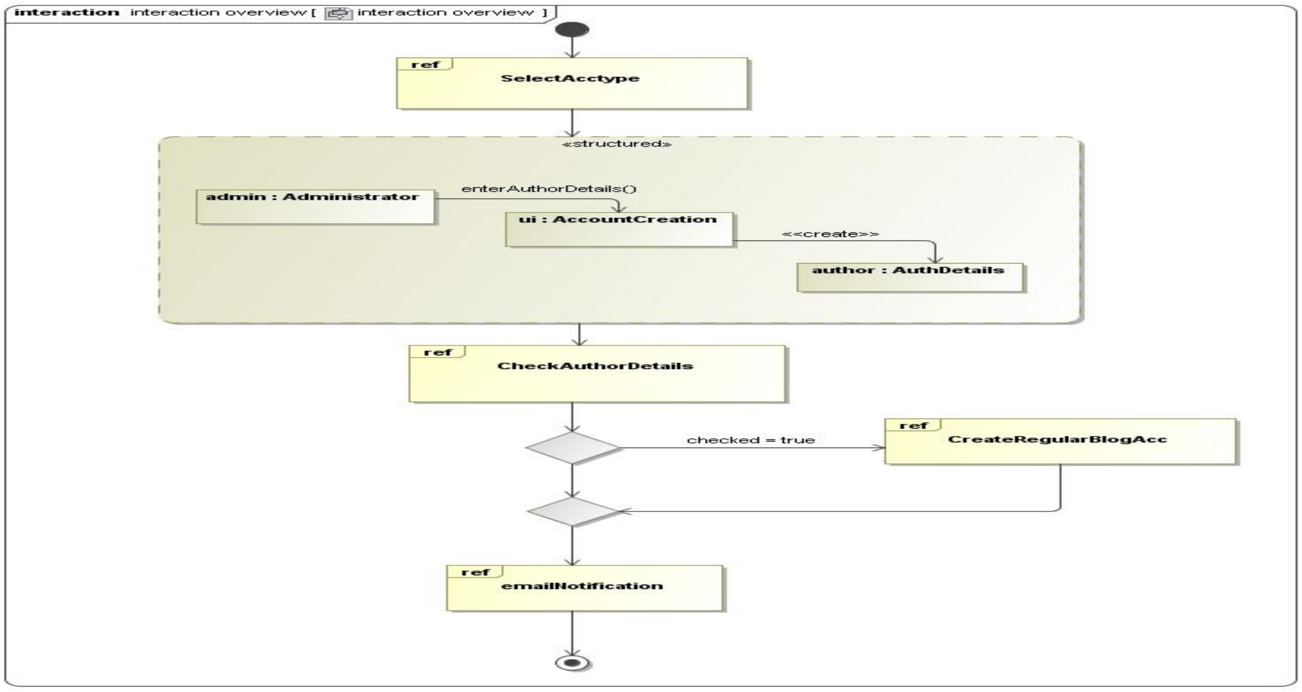


**Fig. 4: Interaction Overview Diagram for creating weblog**

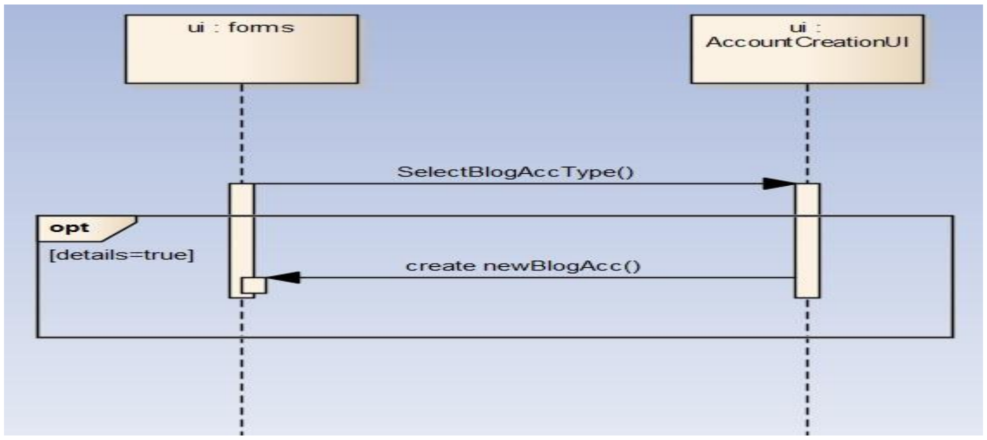


**Fig. 6: Sequence diagram for CreateRegularBlogAcc**

This is where use case generalization comes in. A more common way of referring to generalization is by interaction occurrences in sequence diagram. The most common problem with sequence diagrams is that IN any interaction diagram the redundancy can't be avoided with another sequence diagram i.e. often two scenarios overlap. The solution to the above problem is to make an interaction occurance "SelectAcctype" as shown IN Fig. 4 that can be referred to in several other diagrams as provided in UML 2. Several other operators that can be used in a sequence diagram are optional (opt), repeated (loop), or an alternative (alt) [1].

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*

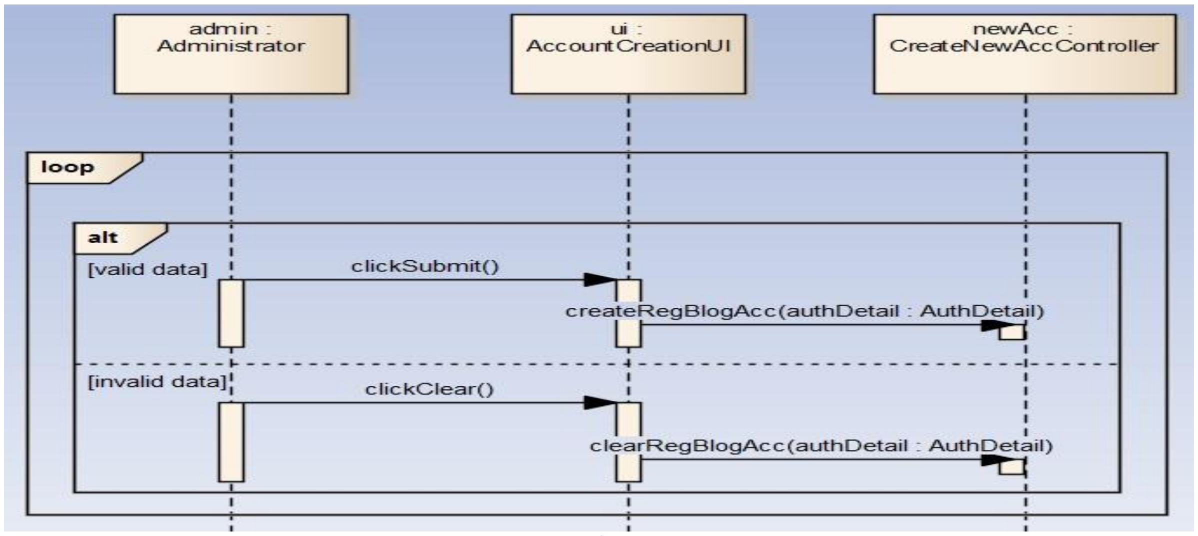

admin : Administrator
ui : AccountCreationUI
newAcc : CreateNewAccController
loop
alt
[valid data]
clickSubmit()
createRegBlogAcc(authDetail : AuthDetail)
[invalid data]
clickClear()
clearRegBlogAcc(authDetail : AuthDetail)

**Fig. 6: Sequence diagram for CreateRegularBlogAcc**

Each of the Interaction Occurrences can be represented by a detailed sub-diagram (sequence diagram) as shown in Fig. 5 and Fig. 6.

These interactions can be combined in different ways to create new scenarios. Sequence diagram can normally represent one scenario; so typically, it is required to use one sequence diagram for the normal scenario and several sequence diagrams for the alternative scenarios. The main purpose of sequence diagrams is to show the order of events between the parts of system that are involved in a particular interaction creating the weblog account.

Communication diagrams add another perspective to an interaction by focusing on the links between the participants or objects. The Administrator selecting an account type and handling the author's details are shown by communication diagram. Communication diagrams explicitly show the links that are needed between participants to pass an intera-ction's messages. With a quick glance at a communication diagram, it is possible to show which participants need to be connected for an interaction to take place. A participant's name formatted as *author: AuthorDetails* represents the *<object>:<class>*similar to participants on a sequence diagram. A *communication link* is shown with a single line that connects two participants. A link from *ui: AccountCreation* to *author: AuthDetails* in Fig. 4 allow messages to be passed between them so that the administrator can store the author details to validate the details entered.

The UML 2 model is being developed using a case tool such as Sparx Systems [21] and the CPN model is designed using CPN tools. CPN models can be integrated with software development process to reduce the hazards of incorrect designs and there by helps in increasing the reliability by incorporating the user controlled view of system simulations. In the case study, a set of users have registered to create a new Blog account shown in the place "*Users*". The CMS updates only if the administrator is available which is shown by constraint that the transition "*content management system"* fires only when token is present in the "*admin"* place. The token named *(n,d)* designating the user-id and nameis passed only whenthe respective transitions fire. The transition named *"CheckAuthorDetails"* is fired only after the respective author details are checked from the place "*valid author"* and further on successful evaluation is added to the database represented by the place *"add to database"*. After blog account has been created for the users they are acknowledged by an Email shown by the transition named *"Email Notification"*. Marking of net is the distribution of tokens in respective places of the net. Derivation of color sets and their variables are defined as shown:

colset ID = int timed;
colset DATA = string timed;
colsetIDxDATA = product ID * DATA timed;
closet UNIT = unit;
closet BOOL = bool with (No, Yes);
funprocessingTime() = discrete(30, 50);

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

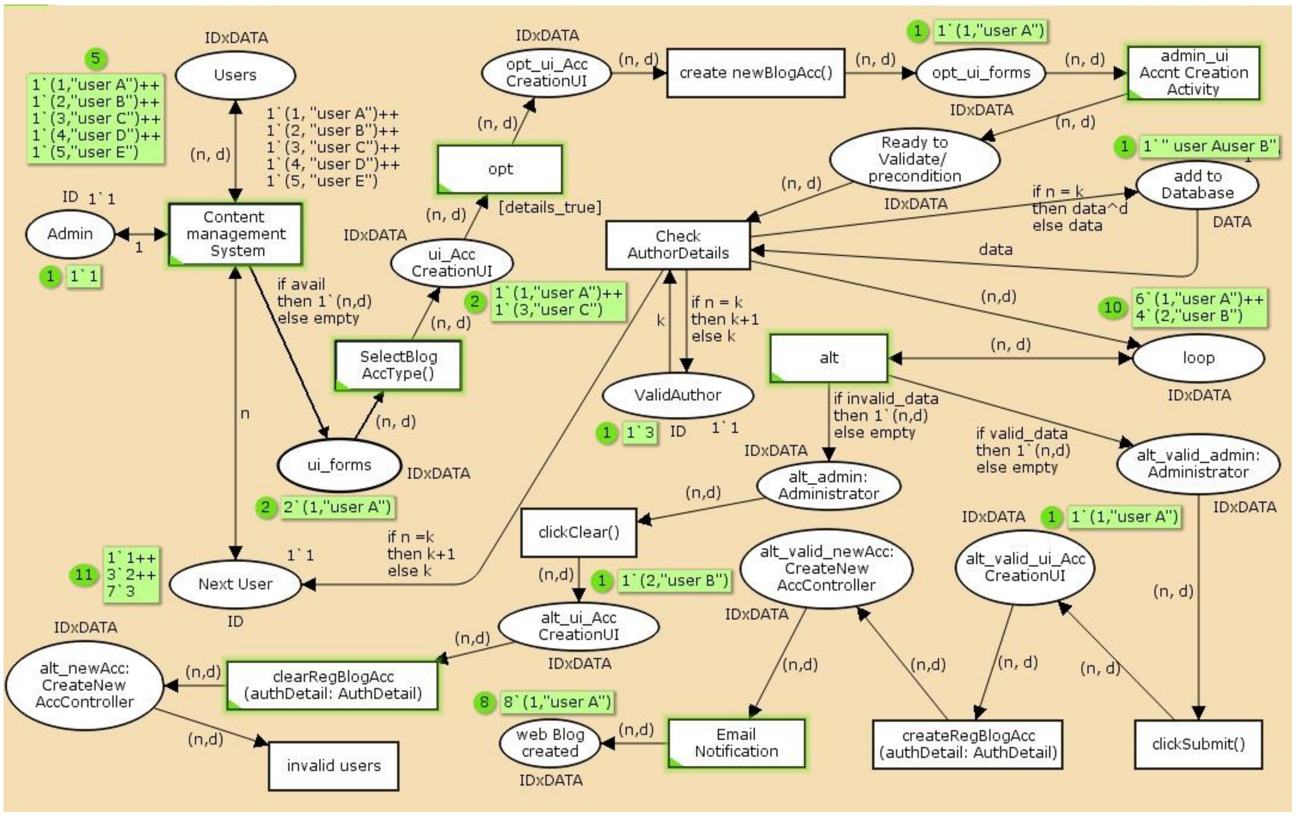


**Fig. 7: Mapped CPN model from interaction overview diagram of Fig. 4**

var n : ID;
varNetworkAvail : BOOL;
var d : DATA;
Here in this example, *"colsetIDxDATA"* is defined as a product of previously defined timed color sets *ID* and *DATA* indicating user-id and name. The variables "*n, k, d*" is used to extract tokens from places and to put new token into output place. The place *"ui_AccCreationUI"* of Fig. 7 is of type *"IDxDATA"* with tokens *"(n, d)"*. The errors caused by network connection can be simulated with the boolean variable *"NetworkAvail"*l associated with the color set *BOOL*.

## VI. RESULTS OF VERIFICATION USING CPN MODEL

The CPN model created after transformation of IOD is executed in CPN Tools. The Movement of tokens from places and the firing of Transitions are closely monitored. A step by step execution pattern of firing the transition is followed to understand the flow of messages in IOD. Verifying the CPN model in Fig. 7, by step by step execution through simulation, some of the errors in the design could be found as follows:. Here from CPN model in Fig.7 it was possible for the same user to have multiple registration for Blog accounts which has to be avoided. While execution we find multiple tokens of "user A" in different *places* like *AccCreationUI*which has 4 tokens of *"user A"*, another 4 set of tokens in place *"Administrator"* after Administrator has checked for authors details.

The users for registration selected from the place "Users" after assigning proper valid user ID is found to lose the sequence for registration for Blog. This may result in long waiting for some users whose ID has already be assigned and still not registered and checked for author details. The no of users to be checked for verification has to be limited to avoid network congestion and delay for author verification. Also it wasthought to add some mechanism to add some privileges for some of the premium users, there by dynamic requirements in the design have to be modelled even during the later stages of design development. We could also find that the model doesn't handle situations if in case of network failure.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

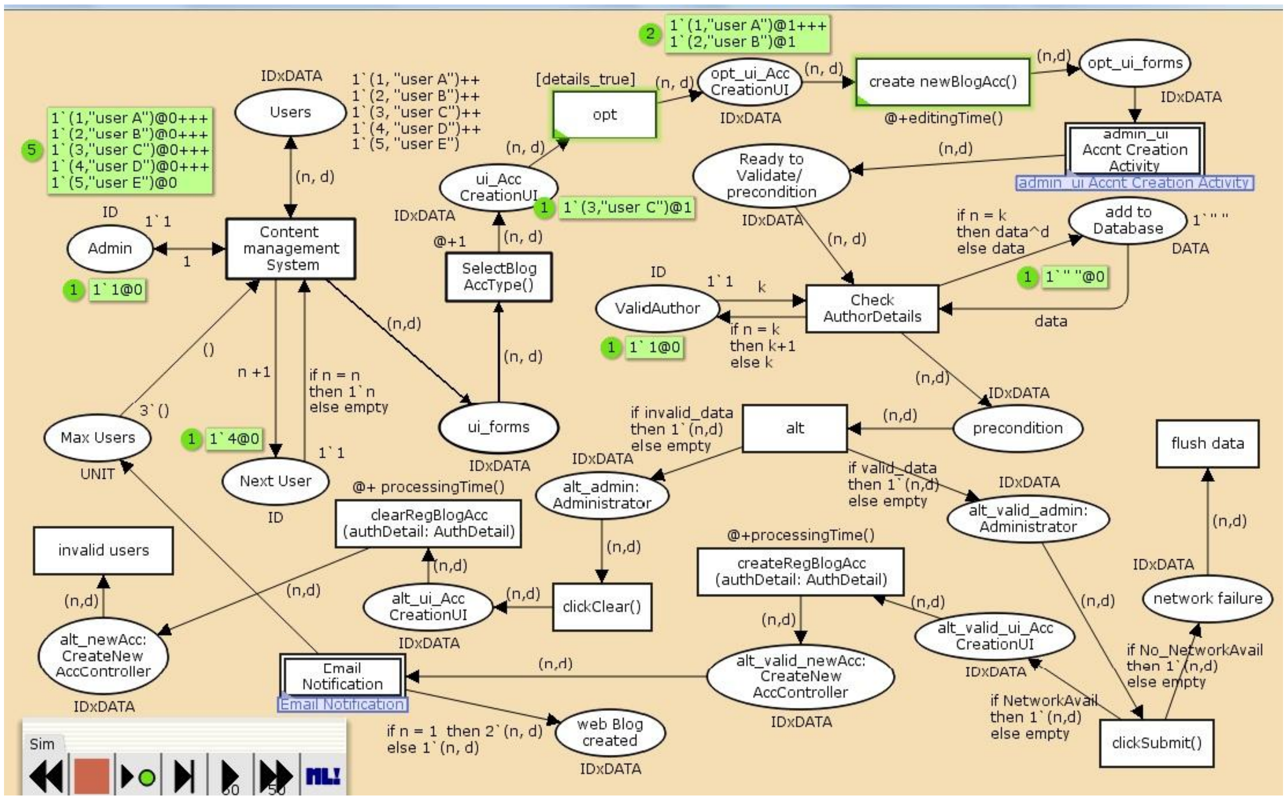

Users
IDxDATA
Content management System
Admin
ID
Max Users
UNIT
Next User
invalid users
alt_newAcc: CreateNew AccController
Email Notification
web Blog created
SelectBlog AccType()
ui_Acc CreationUI
opt
opt_ui_Acc CreationUI
create newBlogAcc()
opt_ui_forms
admin_ui Accnt Creation Activity
Ready to Validate/ precondition
add to Database
Check AuthorDetails
ValidAuthor
ui_forms
clearRegBlogAcc (authDetail: AuthDetail)
alt_ui_Acc CreationUI
clickClear()
alt_admin: Administrator
alt
precondition
alt_valid_admin: Administrator
createRegBlogAcc (authDetail: AuthDetail)
alt_valid_newAcc: CreateNew AccController
alt_valid_ui_Acc CreationUI
network failure
flush data
clickSubmit()
if n = 1 then 2`(n, d) else 1`(n, d)
if invalid_data then 1`(n,d) else empty
if valid_data then 1`(n,d) else empty
if No_NetworkAvail then 1`(n,d) else empty
if NetworkAvail then 1`(n,d) else empty
if n = k then k+1 else k
if n = k then data^d else data
if n = n then 1`n else empty
@+ processingTime()
@+editingTime()
Sim

Fig. 8 CPN model after rectifying the errors from Fig. 7

## VII. RECTIFYING THE DESIGN ERRORS FROM THE MAPPED CPN MODEL

To rectify the errors in the design of system that was identified from the CPN model of Fig.7, a model was further designed in CPN as shown in Fig. 8 with the following details: A place named "*Max Users*" has be added to the CPN model in Fig. 8 to avoid network congestion restricting the no of users in the system. For design simplicity it was made to allow a maximum of only 3 users to create a blog account simultaneously. The concept of timed tokens is included and the arcs from the place "*Next User*" have been modified accordingly to avoid users waiting for long for their account verification. A considerable amount of delay is added for the processing of *"createRegBlogAcc"* transitionusing the function*: "funprocessingTime()" after it was included in the declaration as "fun processingTime() = discrete(30, 50);"*. This is represented by *"@ + processingTime()*" mentioned along with the transition "*createRegBlogAcc*" in Fig. 8. An additional privilege for premium registration ("user A") creating an extra blog account has been provided thereby satisfying dynamic requirements to the design. This is shown by the arc having inscription *"if n = 1 then 2`(n,d) else 1`(n,d)*" to the place "*web Blog created*". Case of network failure at any point of transaction, is handled by arc with inscription "*If NetworkAvail then 1'(n,d) else empty*" from the transition *clickSubmit()*.

## VIII. STATE SPACE ANALYSIS

A possible next phase is to apply the state space tool to verify and validate the functional correctness of the system. This compilation is handled by the simulator part of CPN ML. The first phase of applying the state space tool typically consists of making the CPN model tractable for state space analysis. The next step is then to generate the state space. A part of the state space generated for the CPN model of Fig. 8 is shown in Fig. 9. Each node represents a reachable marking, while each arc represents the occurrence of a single binding element leading from the marking of the source node to the marking of the destination node. The number at the top of each node of state space generated represents the node number. The number of predecessor and successor nodes for each state is separated by a colon. Here in Fig. 9 it is found in node number 17 that @ 1 time instance, there are tokens 1`(2, "user B") and 1`(3, "user C") in the place *CMS` ui_AccCreation_ui*. Binding elements for each state transition can be found on arcs from state space by selecting the arc. The arc from *node 5* to *node 8* indicates the binding as *8: 5→8 @0 CMS SelectBlogAcc_Type 1: {n=2, d="user B"}*. This indicates that the transition *SelectBlogAcc_Type* was fired passing the token *"n=2, d=user B"*.To improve readability, only the detailed contents of one of the markings and some of the binding elements are shown. Strongly connected components facilitate to interpret whether a set of states can be reached from a given state.

In addition to state space graph, a state space report may also be generated for verification of large state spaces. It provides information about the quality parameters of the CPN with properties like Home, Liveness, Fairness etc. These properties aim at identification of final states and transitions that will not fire from an initial marking. The initial part of the state space report contains some statistical information about the size of the state space.

**Statistics for partial state space**

```
---------------------------------------------------------------
State Space
     Nodes:  36748
     Arcs:  57966
Secs:  300
     Status:  partial
Strongly Connected Component
     Nodes:  36748
     Arcs:  48962
Secs:  0

Boundedness Properties
```

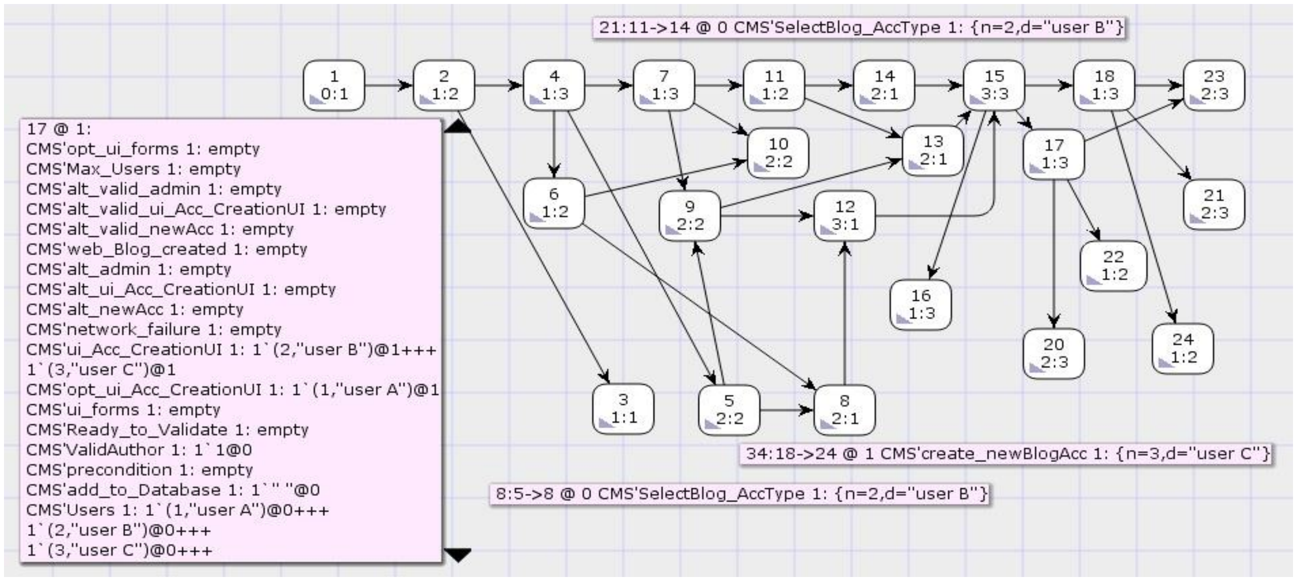
| Best Integer Bounds | Up | Low |
|---|---|---|
| CMS'Users | 5 | 5 |
| CMS'alt_ui_Acc_CreationUI | 3 | 0 |
| CMS'network_failure | 3 | 0 |
| CMS'add_to_Database | 3 | 0 |

```
Best Upper Multi-set Bounds
CMS'Max_Users
3`()
CMS'alt_ui_Acc_CreationUI
1`(1,"user A")++
1`(2,"user B")++
1`(3,"user C")

CMS'Ready_to_Validate
1`(1,"user A")++
1`(2,"user B")++
1`(3,"user C")

CMS'web_Blog_created
1`(1,"user A")++
1`(2,"user B")++
1`(3,"user C")

Home Properties
-------------------------------------
Home Marking: None

Liveness Properties
-------------------------------------
Dead Markings: 18115
Dead Transition Instances: None

Live Transition Instances: All
Fairness Properties
-------------------------------------
No infinite occurrence sequences.
```

The minimum and maximum values of token associated with a place can be shown by Boundedness properties. These properties help in deciding as to how many objects need to be instantiated for a system to meet its requirements. Home markings denote whether any state returns to the initial state or not. Liveness properties help to find any transitions that are not fired throughout the simulation run. Finally fairness properties check for infinitely occurring sequences. Thus a system can be validated by removing any number of invalid states and deadlocks which enables the software analyst to redefine the use cases and conceptual models [19], [20]. CPN tools help the analyst to derive information regarding the number of reachable states and transitions that do not fire in a CPN model. The state space graph generated by CPN tools demonstrates whether the model is profound and complete. It contains every possible sequence of state changes from initial state to the final state. Every path in state space graph should be consistent with the desired behaviour.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 2, Issue 5, May 2013*



```
21:11->14 @ 0 CMS'SelectBlog_AccType 1: {n=2,d="user B"}
34:18->24 @ 1 CMS'create_newBlogAcc 1: {n=3,d="user C"}
8:5->8 @ 0 CMS'SelectBlog_AccType 1: {n=2,d="user B"}
```

```
17 @ 1:
CMS'opt_ui_forms 1: empty
CMS'Max_Users 1: empty
CMS'alt_valid_admin 1: empty
CMS'alt_valid_ui_Acc_CreationUI 1: empty
CMS'alt_valid_newAcc 1: empty
CMS'web_Blog_created 1: empty
CMS'alt_admin 1: empty
CMS'alt_ui_Acc_CreationUI 1: empty
CMS'alt_newAcc 1: empty
CMS'network_failure 1: empty
CMS'ui_Acc_CreationUI 1: 1`(2,"user B")@1+++
1`(3,"user C")@1
CMS'opt_ui_Acc_CreationUI 1: 1`(1,"user A")@1
CMS'ui_forms 1: empty
CMS'Ready_to_Validate 1: empty
CMS'ValidAuthor 1: 1`1@0
CMS'precondition 1: empty
CMS'add_to_Database 1: 1`""@0
CMS'Users 1: 1`(1,"user A")@0+++
1`(2,"user B")@0+++
1`(3,"user C")@0+++
```

**Fig. 9: State space generated for CPN model**

The state space graph generated by CPN tools demonstrates whether the model is profound and complete. It contains every possible sequence of state changes from initial state to the final state. Every path in state space graph should be consistent with the desired behaviour.

## IX. CONCLUSIONAND FUTURE SCOPE

In this paper an approach was explicated to reduce the gap between informal and formal methods of loosely coupled software specification, verification, and validation methodologies. Here a proposal of developing CPN models to validate the IOD sequence fragments and evaluate systems modeled is highlighted. As CPN models are executable, it is possible to investigate the behaviour of the system by making simulations of the CPN model. A case study to create web blog account through content management system allowing an administrator to verify the author credentials database is mapped and verified through CPN model.

Extension of this work may be thought of developing methodology for transformation of concurrent composite state chart diagrams to CPN. Also, the use of timing constraints from timing diagrams and temporal information from sequence diagrams can be eventually annotated for performance evaluation using CPN.

## ACKNOWLEDGMENT

We express our sincere gratitude towards the CSE dept. of Christ University Faculty of Engineeringespecially Prof. Balachandran for his constant motivation and support.

## REFERENCES

[1] S. S. W. Ambler. The Elements of UML 2.0 Style.Cambridge University Press, 2005
[2] ITU-T: (1996). 'Message Sequence Charts (MSC)'International Telecommunication Union. Recommendation Z.120.
[3] H. Störrle and J. H. Hausmann (2005) 'Towards a Formal Semantics of UML 2.0 Activities', Software Engineering, pp. 117-128.
[4] T.S. Staines. 'Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets', 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, Belfast, Northern Ireland, pp.191-200,2008.
[5] V. S. W. Lam. 'On π-Calculus Semantics as a Formal Basis for UML Activity Diagrams', International Journal of Software Engineering and Knowledge Engineering, vol. 18, n°. 4, pp. 541-567,2008.
[6] M. V. Cengarle, A. Knapp (2005) 'Operational Semantics of UML 2.0 interactions', TUM-Report, n°. TUM-I0505, TechnischeUniversitätMünchen.
[7] M.V. Cengarle, P. Graubmann, S. Wagner (2006) 'Semantics of UML 2.0 Interactions with variability', Electronic Notes inTheoretical Computer Science, vol.160, pp.141-155.
[8] A. Knapp, J. Wuttke (2007) 'Model Checking of UML 2.0 Interactions', Lecture Notes in Computer Science, Springer, vol. 4364, pp. 42-51.
[9] L. Kloul and J. Küster-Filipe 'From Interaction Overview Diagrams to PEPA Nets', In proc. of the Workshop on Process Algebra and Stochastically Timed Activities PASTA 2005, Edinburgh, Scotland, pp.7- 8, 2005.
[10] K. Jensen (1992), 'Colored Petri Nets -Basic Concepts, Analysis Methods and Practical Use'.Volume 1: Basic Concepts. Springer-Verlag.
[11] K. Jensen.(1995) 'Colored Petri Nets -Basic Concepts, Analysis Methods and Practical Use'.Volume 2: Analysis Methods. Springer-Verlag
[12] Jensen, K., Kristensen, L. M., and Wells, L 'Colored Petri Nets and CPN Tools for modeling and validation of concurrent systems'. Int. J. Soft. Tools Technol. Transf, 2007, pp. 213-254.
[13] CPN tools, www.daimi.au.dk/cpntools.
[14] J.Cortadella, A. Contradetyev, L. Lavangno, S.Moral. 'Task generation and compile time scheduling for mixed data control embedded software'. In proc. of the design automation conference, 2000.

[15] Bouabana-Tebibel,Thouraya. 'UML2 Interaction Overview Diagram Validation'. Proc. of the 2009 Fourth International Conference on Dependability of Computer Systems, IEEE Computer Society, pp. 11-16.
[16] Bouabana-Tebibel, Thouraya.'Semantics of the interaction overview diagram'. Proc. of the 10th IEEE international conference on Information Reuse & Integration, IEEE Computer Society, 2009, pp. 278-283.
[17] M. V. Cengarle, A. Knapp. (2005) 'Operational Semantics of UML 2.0 interactions', TUM-Report, n°. TUMI0505, TechnischeUniversitätMünchen.
[18] J. Jørgensen and S. Christensen. 'Executable design models for a pervasive healthcare middleware system'. In Proc. of the 5th UML Conference, volume 2460 of Lecture Notes in Computer Science, Springer-Verlag, 2002, pp. 140–149.
[19] J. Campos and J. Merseguer. 'On the integration of uml and petri nets in software development'. In 27th Int. Conf. on Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2006), volume 4024 of Lecture Notes in Computer Science, Springer, pp. 19–36.
[20] Andrade, Ermeson and Maciel, Paulo and Callou, Gustavo and Nogueira, Bruno. 'Mapping UML Interaction Overview Diagram to Time Petri Net for Analysis and Verification of Embedded Real-Time Systems with Energy Constraints'. In Proc. of the International Conference on Computational Intelligence for Modeling Control & Automation, IEEE Computer Society, 2008, pp. 615-620.
[21] Sparx Systems, www.sparxsystems.com.
[22] Vinai G. Biju and S. K. Rath. 'CPN Tools as a Supplement to UML for Validation of Software Requirements", Proc. of 4th National Conference,IndiaCom, 2010.

## BIOGRAPHY

**Vinai G. Biju** received his Bachelors in Engineering degree in Computer Science and Engineering from Institution of Engineers India in 2008 and Masters in Technology in Computer Science and Engineering from National Institute of Technology (N.I.T) Rourkela, India in 2010. He has worked in Accenture in the area of Data warehousing. He is now working as an Assistant Professor in the Department of Computer Science and Engineering at Christ University Faculty of Engineering, Bangalore, India. His main research interest includes MDA, formal methods and model checking.

**Vinod. K. Agrawal** received his Ph.D. from Indian Institute of Sc., Bangalore, India He was the Group Director for Control Systems Group, ISAC at Indian Space Research Organisation (ISRO) and was also the project director of GSAT-4 at ISRO, India. He is now the director of Crucible of Research and Innovation and Professor in the Department of Information Science and Engineering at PESIT, Bangalore, India. He was awarded Astronautical Society of India Award Govt. of India, Dept of Space Merit Award, IETE-HariRamjiToshniwal Gold Medal Award. He has more than 80 papers in research journals and refereed conferences. His main research interests include formal methods in software engineering, fault-tolerant and distributed computing He is a senior member of IEEE. He is also a member of International Society for Hybrid Micro-electronic and a member of the Astronautical Society of India.