Formalization of UML Composite Structure using Colored Petri Nets

Rao Sohail Iqbal\textsuperscript{1}, Ramzan Talib\textsuperscript{2}, Haseeb Ur Rehman\textsuperscript{4}
Department of Computer Science
Government College University
Faisalabad, Pakistan

Muhammad Awais\textsuperscript{3}
Department of Software Engineering
Government College University
Faisalabad, Pakistan

Wajid Raza\textsuperscript{5}
Department of Computer Science
NCBA&E Lahore, Multan Campus
Multan, Pakistan

Abstract—Design specification and requirement analysis, during development process involved in transformation of real world problems to software system are subjected to severe issues owing to involvement of semantics. Though, for design and specification of object-oriented systems, Unified Modeling Language (UML) is now recognized as standard language however, its structures have numerous drawbacks which include lack of semantics definition and unidentified deadlocks. The research work proposes a model to avoid deadlocks, specifically in composite structure of UML. Verification of system models by formal methods holds significance, particularly, at requirement specification and design level, to ensure the accuracy of models and high light the design problems before implementation. The paper proposes the rules that allow software engineers to formalize the behavior of UML 2.0 composite structure using Colored petri nets. Using these rules, the research shall analyze the correspondent Colored petri nets and conclude the properties of the original work flow, using theoretical outcomes in the Colored petri nets domain.

Keywords—Design specification; UML (Unified Modeling Language); semantic; transformation; deadlocks

I. INTRODUCTION

Software engineering is facing multifarious challenges in the fields of design specification and requirement analysis owing to complications involved in system models’ verification of real world thus rendering it a difficult task [1]. UML, governed by Open Group of Companies, is comparatively open standard, organized by the Open Management Group (OMG). It helps to make all important decisions of specification related to analysis, design and implementation. Though it is not visual programing, but its models are connected to a huge variety of programming languages [2]. It plays a vital role in design and implementation segment for building a software system. However, combination of a variety of object-oriented analysis and design methods into a single modeling language has inherited complicated issues [3]. Flaws in UML notation have invited the researchers to discover alternative methods for addressing the integration and compatibility issues with in order to allow an accurate and reliable design, modeling and development. UML has a set of standard symbols required for building objects and visualization of a software system, non-software systems and business modeling [4]. It has become a standard for designing and implementation of object-oriented system [5], even though its semantics are semi-formal and liable to chances of ambiguities in the system design. Issues related to Modeling a system by UML are as under:

- Due to the graphical notation of UML structure it has certain chances of errors.
- At the design level of software system, hidden semantics of UML diagram can cause ambiguities.
- Model described by UML diagram can lead to multiple understandings, thereby disabling receivers of the model to take the precise decision with respect to the diagram.

The capability of UML can be enhanced by connecting it with formal methods and design the system by describing semantic rules in a formal way [6] owing to following reasons:

a) Formal methods provide high level guarantee and reliability for analyzing of models.

b) It helps in detection of errors and defects at an early stage to reduce the cost.

c) Formal methods allow the developer to check model mathematically and prove its validity by the integration of UML diagrams with formal notations resulting in reliable, complete and accurate modeling of the system [7] [8].

UML 2.0, comprising integrated models and diagrams was introduced after success of UML 1.x. It allows us to model the structure of a system before its implementation, however significant problems related to identification of “errors” and “deadlocks” in designing phase persist [9] thereby leaving chances of system failure after execution. This research aims at resolution of this problem by transforming composite structure, using formal methods. Composite structure diagram enables us a deep insight of the class so as to what is actually happening as well as relationship of nested classes [10]. Using Colored petri nets model for transformation process which is a mathematical modeling (formal verification) language used for systems which have complex behavior. Since Colored petri nets are a formal model so they do not carry any ambiguity and can be validated [11]. The prominent benefit of this language which compelled us to deviate from traditional UML to Colored petri nets model is that it identifies the ‘Errors’ and ‘Deadlocks’ in a system before execution [9]. This will help to evaluate the model at an early stage thereby reducing the cost.
risk and time span [12]. In the past few years a lot of work
gone into formalization of UML diagrams, however composite
structure diagram has failed to attract any worthwhile strength
of researchers in the field. By transforming composite
structure using Colored petri nets model will try to attain
much precise results, while proving the scenario using some
rules and formulate a case study towards the end.

II. RELATED WORK

A combination of Z notation and petri nets is proposed for
designing safety critical systems. The part of construction cell
is modeled and check its validity by safety interrelated
properties [13]. A transformation from OCL (Object
Constraint Language) into B is projected in [14]. Given
transformation scheme is automatically derive to B except
invariants of class. A UMC framework is present by the
gatherings of UML state machine for the formal verification
of concurrent systems. The formal model which is given in
this system have transition having labels dually and uses logic
to describe its properties comprising event-based and state-
based logic. It allows user to see insights of a UML model to
investigate behavioral aspects by visualization and model
checking [15]. A framework of fuzzy logic is suggested to
evaluate the compatibility of formalism on specific electronic
commerce system. The evaluation of UML, Z notation, state
charts, finite state machine (FSM) and Colored petri nets is
carried out in combination with some domain by using the
fuzzy logic structure and evaluation standards [16].

Raymond [17] shows the growing diversity of formal
methods and mathematical models which plays a bridge role
between “continuous” and “discrete” systems. A method is
proposed by Than et al. [18] to formalize the syntax and
semantics of UML state chart into Z notation, given semantics
helps to elaborate the model consistency and assure its
unambiguity and completeness. A formalization of higher
order logic is presented based on arithmetic by using tool
SPW (Signal Processing Work System) [19]. Akbarpour
uses the different rounding modes in fix point arithmetic like
directed and even rounding modes. An analysis is performed
to check the accuracy and basic arithmetic operations,
multiplication, subtraction, addition and division.

Liu and Dong [20] give an algorithm to schedule the tasks
executed in the finite set of memory. Input streams and code
generated will be smaller in size as compared to previously
used. Petri nets is used to do formal analysis and verification
of the given algorithm. A simulation is projected by Hasan
and Tahar [21] a higher order logic theorem for the analysis of
critical parts and can give accurate results. A pair of
probabilistic properties is also verified which was not
evaluated by existing techniques. A methodology is offered to
formalize some common diagrams of UML which are used in
several phases of Z notation software development [22], a
visual representation has been taken with this tool based
approach by the formalization of UML diagrams. Derakhshandeh et al. [23] proposed a formal model for Access
Control Policies (ACP) specification which have ability to
express many ACPs, joining them in a unified model and also
verifying conflicts among them.

Ma [24] presents the fuzzy extensions of ER/ERD and
UML models to manage a system at conceptual level. Some
main notations have been prolonged and conforming the
graphical notations, a formalization is performed by aiming on
the basic concepts of class structure using simple case studies.
A formal method is proposed based on mathematical
mechanism to prove program correctness [25]. Algebra
approach is used to improve the verification capability of the
model which give the guarantee to provide error free model.
Cunha et al. offered a method for transformation of UML
sequence diagram to Petri nets to find out dead locks, errors
and verifying liveness, reachability and safety of the model.
This method is performed in an embedded control application,
a sensory device which detect the object in the environment. A
mathematical extension of OCL language is projected by
cumulate its class library to deploy some mathematical notions
including relations and functions [26]. Zhang and Liu [27]
give a web-based service CCML (Cooperative Composition
Modeling Language), is formalized by the formal verification
method and also check its validity by proving a case study. A
target fusion recognition method is proposed to check its
reliability and validity based on the fuzzy sets and Petri nets
under a complex environment which is also demonstrated by a
simulation example [28]. UML activity model does not have
firmly formal semantics, so it is hard to analyze them and
activity model checking. An ontology-based method is
presented for semantic checking of activity model which are
divided into two parts dynamic semantics and static semantics
[29].

III. BACKGROUND

In this section a brief discussion has been presented about
the composite structure diagram and Colored petri nets.

A. Composite Structure

In UML 2.0 many diagrams were introduced which
comprises Activity diagram, Class diagram, Sequence
Diagram, Communication diagram, Component diagram,
Composite Structure diagram, State diagram, Agent based
UML and so on so forth. Composite structure diagram shows
the internal structure of a specific classifier just like
component diagram. A huge range of the notation of
component diagram can be supported by composite structure
diagram as well. Where the component diagram shows the
internal structure, on the other hand composite structure shows
the specific functionality of the classifier to be executed. It has
four main parts which includes class, connectors, ports and
interfaces. Class shows all the information, and composite
structure diagram assumes to be complete when class diagram
is considered as the smallest class structure which declares the
given composite structure [30].

A connector is providing the interface between two
objects. It might be an object which represent the relationship
or a value hold in the variable. The value which is hold by the
variable is the other way to represent the association between
two objects. A port is a source which suggest the functionality
of composite structure without exposing the internal
operations. The port is a small square notation which is drawn
above the edge line of the boarder. The multiplicity of port
and its parent name are written near to it. The ports are

www.ijacsa.thesai.org
connected with required/provided interfaces like from/to the situation. The required/ provided interfaces are commonly show with the socket/ball notation. Each port is connected to its internal operation using connectors and these internal operations can be some code kept by the classifier. The composite structure diagram and associated structures uses and semantics are well defined in [31][32][33] while the concepts of composition are effectively defined in [34][35].

B. Petri Nets

Petri Nets is a formal mathematical modeling language which is introduced in 1962 by Carl Adam Petri. It is used to formalize the models before implementation to identify the errors and deadlocks hence it can be rectified. It has strong formalized modeling capability in various fields like computer science and system engineering. It gives us facility to provide the mathematical theory with pictorial representation for a dynamic system. Mathematical theory will draw a precise model and let us know the behavior of system. And the graphical system shows the model visuals and state changes of the system. The combination of these two aspects are the key reasons of petri nets success. It has been used in modeling of many event driven systems such as computer networks, work flows, communication systems, real time computing systems, manufacturing plants, logistic networks and command and control systems [36].

Petri nets are the directed split graph having transitions and places which are represented by the nodes. Transitions are the events which may be occur and it is denoted by the bars and places are the conditions denoted by circles. Petri nets are consisting of three main parts which include places, transitions and arcs. Arcs are directed from places to transition or transition to places. They will never connect from place to place or transition to transition. The arc which comes out from place to transition that place will called transition input place, and arc which connects from transition to place is called transition output place. In the diagram of a petri nets, places are denoted by circles, transition are by long narrow rectangle and arcs are symbolized by one-way arrow which represent the relation from place to transition and transition to place [37].

IV. COMPOSITE STRUCTURE TO COLORED PETRI NETS MODEL FORMALIZATION RULES

In this segment, shown a briefly description about the transformation and formalization of some upper level operators which are offered in the UML2.0 composite structure into interactively equal Colored petri nets. For this purpose, here describe the semantics of operators, and explain how formalization will be accomplished. Furthermore, presented some outcomes by applying these concepts on a few demonstrative samples and also explain by a case study.

A. Component

In rule 1, shown the trivial case where component A is implies the existence of class A. Now show its obtained colored petri nets fig. 1.

B. Simple Composite Structure with Class into Component

The case in rule 2 is denoted the situation of a simple composite structure A contains the B parts. This implies that class A having a sort of relationship with class B. When the relationship is executed, objects may communicate with each other by calling their operations. The part B is shown by solid rectangle with the composition of part A, though the relationship of A and B must be composite. A strong relationship shown by black diamond, which means that part B is life time owned by the parent object part A. Obtained CPN are shown in fig. 2.

C. Component with Port

In composite structure a connection is made by port artifacts and signals send and receive via these ports. A trivial case of composite structure with port is shown in rule 3, which is also implied in class diagram. Connection between ports is termed as “wire”, elaborate that a communication is occur by a signal sent from “wire” to receive port. Obtained Colored petri nets are given in fig. 3.
Rule 3. UML Composite Structure.

Fig. 3. Obtained CPN.

Rule 4. UML Composite Structure.

Fig. 4. Obtained CPN.

D. Two wired Ports

In this rule port p of A is by default visible and public to internal and external parts, although it is the parent class of B and C and port q and r visibility is internal and private because they are the child classes of A. Let us check its obtained CPN in fig. 4.

E. Various Interface Combination

If the interfaces are compatible, then two ports can communicate sensibly between required interface and provided interface. In rule 5 it is possible to establish connection in both directions for ports p and q as each either provided interface or required interface I and J. If connection for instance, between port p and port r are not sensibly connected to each other, no message will be understood and propagate with each other. And a meaningful communication will not take place as neither the required nor provided interface match. Obtained CPN are shown in fig. 5.
The obtained petri nets elaborate the various connecting interfaces of a composite structure, where B is a class having both provided and required interface. After transition of class B, another interface connecting the class C with provided interface.

V. CASE STUDY

By taking a real-world case study of a travel agency shown in fig. 6. We will prove the composite structure diagram by explaining each of its component. For this purpose, formalize a composite structure diagram into Colored petri nets model.

By taking a real-world case study of a travel agency formalization of composite structure will explain more clearly. In this case study proposed a structure of a travel agency that how it works with Web Agent, Agency Offers module and Internal Authentication. Furthermore, show the interactions between customer and agency also their relationship.

A. Web Agent Module
- There are two components to choose in the ‘Web agent’.
- One is ‘Customer app’ for customer only and second is ‘Agency app’ for agency only.
- Each component has its connecting port with ‘Required interfaces’.

The formalized web agent module is shown in fig. 9.

B. Agency Offers Module
- There are four classes to access in the ‘Agency offers’.
- One is ‘Catalog’ for agency catalog information and second is ‘Agency’ for Agency detail.
- Third is ‘Travel’ for getting information about services of travelling and fourth is ‘Availability’ to check the availability of flights.
- Each class and component have its connecting port with ‘Provided interfaces’

The formalized agency offers module is shown in fig. 8.

C. Internal Authentication
- Before confirmation of flight an Internal Authentication of passenger is required after completing all formalities.

VI. FORMALIZED COLORED PETRI NETS MODEL

A formalized colored petri nets model has been given below which shows the different operations performing in the model. In this model presented a web agent module taking the queries of the customer and forward it to the agency offers module and after internal authentification of the customer the process will place towards the final proceedings. Fomalized model is shown in fig. 7.
Fig. 7. The Agency Page – Architecture of CPN Model with Initial Marking $M_0$. 

\(\text{Abt}_\text{Pkgs} \rightarrow \text{Abt}_\text{Comny} \rightarrow \text{Travels} \rightarrow \text{Abt}_\text{Destiny} \rightarrow \text{Stats} \rightarrow \text{STATUS} \)

\(\text{Cust}_\text{Required}_\text{Services} \rightarrow \text{Provide Services} \rightarrow \text{Authenticate} \rightarrow \text{Internal Authentification} \rightarrow \text{Web}_\text{Agent} \rightarrow \text{Response} \rightarrow \text{Custmr} \)

1"CLEARED"++
1"NOT-CLEARED"

1"Akram"++
1"Sajid"
Fig. 8. The Agency Offers Module.
Fig. 9. The Web Agent Module.
VII. CPN TOOLS STATE SPACE REPORT

CPN Tools state space report for:
/cygdrive/F/2 AFTER PhD 23.4.2018/RESEARCH PAPERS/GC FSD PAPER/TicktingHier.cpn
Report generated: Sun May 13 13:56:09 2018

Statistics
-----------------------------------------------

State Space
Nodes: 20039
Arcs: 99589
Secs: 300
Status: Partial

Scc Graph
Nodes: 20039
Arcs: 99589
Secs: 2

Boundedness Properties
-----------------------------------------------

Best Integer Bounds

<table>
<thead>
<tr>
<th>Property</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency'Abt_Comny 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Abt_Destiny 1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Abt_Pkgs 1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Authenticate 1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Agency'Cust_Required_Services 1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Custmr 1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Response 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Agency'STATUS 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Stats 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Agency'Travels 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AgencyOffers'Avail 1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AgencyOffers'Company_Info 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AgencyOffers'Destination 1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>AgencyOffers'Packages 1</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>WebAgent'Custmr_LIST 1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Best Upper Multi-set Bounds

<table>
<thead>
<tr>
<th>Property</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency'Abt_Comny 1</td>
<td>1&quot;Pakistan Travels&quot;</td>
<td></td>
</tr>
<tr>
<td>Agency'Abt_Destiny 1</td>
<td>1&quot;ASIA&quot;++</td>
<td></td>
</tr>
<tr>
<td>Agency'Abt_Pkgs 1</td>
<td>1&quot;(1,&quot;USA&quot;)++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1&quot;(2,&quot;USA&quot;)++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1&quot;(3,&quot;UAE&quot;)++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1&quot;(4,&quot;UAE&quot;)++</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1&quot;(5,&quot;EUROPE&quot;)++</td>
<td></td>
</tr>
</tbody>
</table>
AgencyOffers'Packages
  1`'(1, "USA")++
  1`'(2, "USA")++
  1`'(3, "UAE")++
  1`'(4, "UAE")++
  1`'(5, "EUROPE")++
  1`'(6, "EUROPE")++
  1`'(7, "AUSTRALIA")++
  1`'(8, "AUSTRALIA")++
  1`'(9, "ASIA")++
  1`'(10, "ASIA")
  WebAgent'Custmr_LIST
    1`"Akram"
    1"Sajid"

Best Lower Multi-set Bounds
  Agency'Abt_Comny 1 empty
  Agency'Abt_Destiny 1 empty
  Agency'Abt_Pkgs 1 empty
  Agency'Authenticate 1 1"CLEARED"++

1"NOT-CLEARED"
  Agency'Cust_Required_Services 1 empty
  Agency'Custmr 1 empty
  Agency'Response 1 empty
  Agency'STATUS 1 empty
  Agency'Stats 1 empty
  Agency'Travels 1 empty
  AgencyOffers'Avail 1 1"AVAILABLE"++

1"NOT-AVAILABLE"
  AgencyOffers'Company_Info 1 empty
  AgencyOffers'Destination 1 empty
  AgencyOffers'Packages 1 empty
  WebAgent'Custmr_LIST 1 empty

Home Properties
----------------------------------------------------------

Home Markings
  None

Liveness Properties
----------------------------------------------------------

Dead Markings
  13571 [9999, 9998, 9997, 9996, 9995,...]

Dead Transition Instances
  Agency'Internal_Authentification 1
WebAgent"Response_to_Customer 1

Live Transition Instances
None

Fairness Properties

No infinite occurrence sequences.

VIII. CONCLUSION

In this paper presented a bunch of rules and also perform a tool implementation to formalize the composite structure into equivalent colored petri nets. In UML 2.0, composite structure shows the internal structure of the classifier and give a deep insight view and cannot identify the errors and deadlocks with this. Proposed work explores the ways to identify the hidden errors and deadlocks at abstract level. This work is done with advance notation of petri nets. Colored petri nets gives more precise formalization. Hence the rules allow the formation of Colored Petri nets that shows various behavior of components in more accurate way.

REFERENCES


