ISSN:2231-0711
www.ijcset.net

Modeling of Access Control System using Petri-nets

Sunita Kumawat[1], G.N.Purohit[2]

[1]*Amity University,Haryana, Manesar, Gurgaon, India*
[2]*Centre for Mathematical Sciences,*
Banasthali University, Banasthali (Rajasthan), India
[1]ksunita86@gmail.com, [2]gn_purohitjaipur@yahoo.co.in

***Abstract*—In this paper we purpose a Petri-net-based method for** describing the model of an **object-oriented design of Access Control System (ACS), by specifying the object interaction scenarios as Petri nets with an Augmented Marked Petri Net (AMPN) structure. After synthesizing these scenarios into an integrated net, we analyze the system based on the special properties of AMPN. For unique representation of events and conditions in an object-oriented ACS, an improved algorithm is applied to the integrated labelled Petri net of ACS to eliminate duplicate condition labels and event labels, with preserving the event firing sequences. Object-based behavioural specifications of ACS are then obtained as projections of the integrated labelled Petri nets onto the objects. For the illustration we present the model of Office access control system in any academic or industry.**

***Keywords*—Access control System; Petri net; Augmented marked Petri net Graph; Resources; Siphon.**

I. INTRODUCTION

In the past two decades, object orientation has been an influential discipline in software engineering. According to the principles of object orientation, an object is an entity that encapsulates states and behaviours. A system is considered as a collection of objects which are interacting with others in order to accomplish the system functionalities. It can be abstracted in two aspects (structure and behaviour) and two levels (intra-object and inter-object) [13], [14], [18], [19], and [25]. Structurally, objects with the same attributes are grouped into classes while classes having common attributes are generalized to form an inheritance hierarchy. Objects exhibit different behaviour on interacting with others, thus demonstrating different object interaction scenarios.

In the previous literature, there are only a few approaches or methods for deriving object-based behavioural specifications are given set of use cases or object interaction scenarios [12], [14], [15], [16]. These use cases are elaborated and expressed in terms of object interaction scenarios and specified as UML sequence diagrams and collaboration diagrams [6], [7], [12], [15], [16], and [17]. Bordeleau proposed an approach which takes a traceable progression from use cases to the object-based state machines [10], [11]. Dano proposed an approach where the use cases are synthesised into a system design according to some mapping rules [2], [3]. However, these approaches solve only trivial issues. The system design cannot be rigorously analysed for its liveness, boundedness and reversibility. Moreover, they are themselves incomplete and insufficient in the sense that the derived object-based state machines may not reflect exactly the given use cases or object interaction scenarios. On the other hand, there are so many approaches or methods which derive a system from a given set of event traces or sequences. Graubmann proposed a method for constructing an elementary net system from a set of event traces [28]. The method is based on the dependence relation between events. A set of possible states and state transitions, which are compatible to the dependence relation, are deduced. Smith proposed a method for constructing a condition-event system from a set of occurrence nets based on the concept of quotient nets [8]. Hiraishi proposed a method for constructing a Petri net from a set of firing sequences [26]. In Hiraishi's method, a language is first identified from the firing sequences. Based on the dependency relation extracted from the language, a safe Petri net is created. Lee also proposed an approach for integration of use cases using constraint-based modular Petri nets [33]. However, without concepts of object-orientation, these approaches and methods cannot be applied to object-oriented system design.

There are some behavioral aspects of objects which are required to analysis the behavioral properties of Access control system.

Specification constructs for object interaction scenarios being too primitive. Conventional specification constructs for object interaction scenarios lacks the formality for representing the pre-conditions and post-conditions for each event occurrence. These are however essentially required in deriving the object behavioural specifications, where the conditions, events and their causal relationships need to be explicitly specified.

Different abstractions between intra-object lifecycle and inter-object interaction. It is difficult to derive individual object behaviours (within a single object lifecycle) from the object interaction scenarios (among multiple objects) because of the difference in abstraction (intra-object versus inter-object). In the literature of object-oriented system design, there is a lack of systematic approaches to solving this problem satisfactorily.

Difficulty in verifying the correctness of the object behavioural specifications. The object behavioural specifications so derived should be correct in the sense that they reflect exactly the given object interaction scenarios [27], [29]. Without a formal method, one needs to go through all possible object interaction scenarios to ensure correctness of the specifications. The process is time-consuming.

Lack of rigorous methods for analysing the system properties. One major objective in system design is to obtain a live, bounded and reversible system - liveness implies freeness of deadlocks, and boundedness implies absence of capacity overflows, while reversibility refers to recoverability. Without a rigorous analysis method, it is difficult for one to analyze whether the outcome system design is live, bounded and reversible.

In this paper, based on Petri nets, we propose a method for refining Access control system with a given set of object interaction scenarios into object-based behavioural specifications, where the above-mentioned problems can be resolved effectively. It involves the following steps:

Step1. Each object interaction scenario is specified as a labelled Petri net (Labelled Petri Net) with an AMPN-structure (i.e. structurally an Augmented Marked Petri Net).

Step2. The Labelled Petri Nets are synthesised into an integrated net which serves to represent the system. Based on the properties of AMPN-structure, the system is analysed.

Step 3. Duplicate labels are eliminated from the integrated net, while preserving the firing sequences (event sequences).

Step 4. Individual object-based specifications are obtained as projections of the integrated net onto the objects.

Fig. 1 shows an overview of the proposed method.

Our proposed method offers a number of distinctive features.

Formal specification of object interaction scenarios. The object interaction scenarios are specified as unambiguous and semantically rich Labelled Petri Nets. The partial orderings of events as well as the causal relationships between events and conditions are explicitly represented.

Effective analysis on the essential system properties. The integrated system possesses a AMPN-structure. By making use of the special properties of AMPN-structure, the system can be effectively analyzed on its liveness, boundedness, reversibility and conservativeness.

Correctness of the derived specifications. Individual object behavioural specifications are rigorously derived from the object interaction scenarios through synthesis and projection. The specifications so obtained reflect exactly the given object interaction scenarios.

Readiness for implementation purposes. In the specifications, every condition or event is uniquely represented so that they can be readily used for implementation purposes.

The rest of this paper is organized as follows. Section 2 provides the preliminaries to be used in this paper. Section 3 introduces the AMPN-structure and its properties are described. In Section 4, we consider object interaction scenarios as Labelled Petri Nets (Step 1 of the proposed method). In Section 5, we focus on synthesizing the Labelled Petri Nets into an integrated system, and analyzing the system properties (Step 2 of the proposed method). Section 6 includes an algorithm for eliminating duplicate labels from the integrated net (Step 3 of the proposed method). Section 7 depicts how individual object-based behavioural specifications are obtained as projections of the integrated net (Step 4 of the proposed method). Section 8 gives a labelled integrated net of office access control system and its projection onto the objects. The conclusions are included in Section 9.

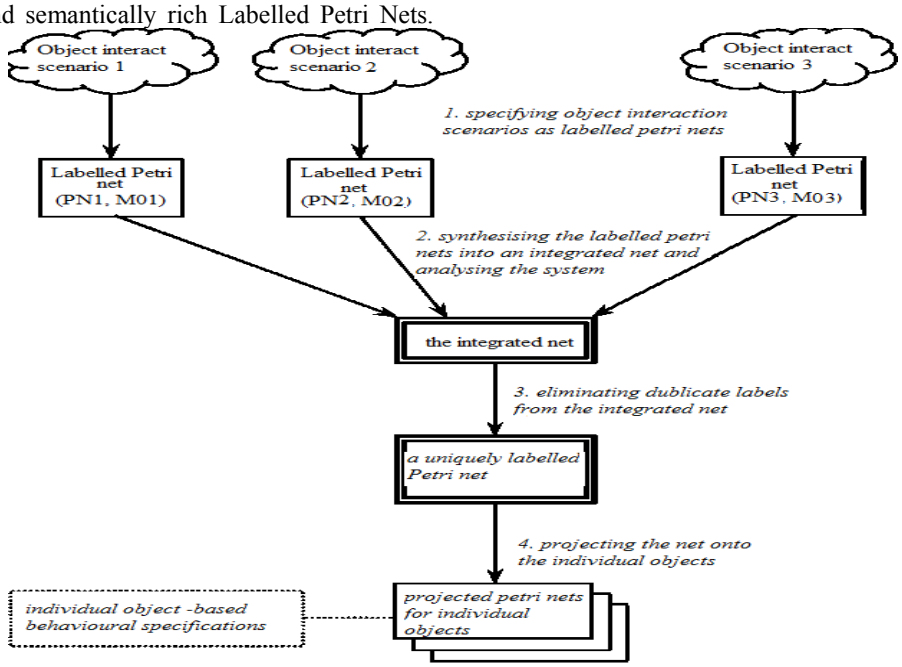

Fig. 1: Overview of the proposed method.

II. DESCRIPTION OF PETRI NET

A *Petri net* is a graphical and mathematical modeling tool for describing and simulating the dynamic and concurrent activities of systems. Petri nets were invented in 1962 by Carl Adam Petri [9]. *A* Petri Net also called place-transition net (*PT*-Net), is a particular kind of directed graph together with an initial state called the *initial marking*. In general a Petri Net is an underlying graph of any directed graph, which is in essence a directed bipartite graph with two types of nodes called *places* and *transitions.* In the Petri Net graph a place is denoted by a circle, a transition by a rectangular box or a bar and an arc by a directed line. The arcs are either from places to transitions (output of places) or from transitions to places (input of places). In other words, the information passes from a place to a transition or from a transition to a place. A Petri Net is a *PT*-Net with tokens assigned to its places denoted by black dots, and initially the token distribution over its places is done by a marking function denoted by M_0. A *token* is interpreted as a command given to a condition (place) for the firing of an event (transition). An enable event may or may not fire, even if all input conditions are fulfilled. The firing of an event or transition changes the token distribution in the places. When a transition fires, it consumes the tokens from its input places, per forms some processing task, and places a specified number of tokens into each of its output places. Fig. 1 illustrates the components and behaviors of an ordinary Petri net. It also shows the transference process from the initial state to the final state by the action of firing. A transition is said to be enabled or fireable at M_0 if for all input places of any transition have at least one token. A transition may fire if it is enabled. The new marking is obtained by removing one token from each of its input places and by putting one token in each of its output places.

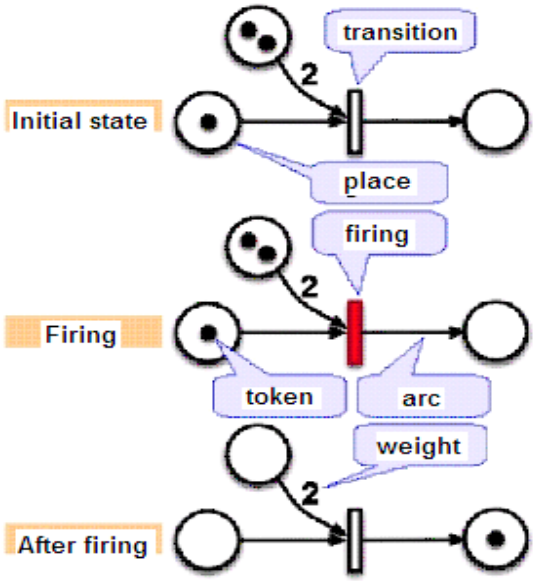

Figure. 1 Petri net model

More detailed and formal description of Petri Nets is given in [1, 4, 5, 9, 24, 30, 32, 34, 37, 38]. We include here some basic definitions, which are relevant to this paper.

Definition 2.1[37, 38]: A place-transition net (PT-Net) is a quadruplet PN = ⟨P, T, F, W⟩, where P is the set of places, T is the set of transitions, such that $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs and W: F → {1, 2 ...} is the weight function. PN is said to be an ordinary PT-Net if and only if W: F→ {1}, we ignore weight one generally in model representation.

A *marking* is a function M_0: P→ {0, 1, 2...}, which distributes the tokens to the places initially. Here M_0 (p) is a non-negative integer associated with place p and represents the number of tokens in the place p at initial marking M_0. M_0 (p) is less than or equal to the capacity of the place p, where capacity of the place is defined as the capability of having the maximum number of tokens at any reachable marking M from M_0. A marking M_0 is said to reachable to M, if there exists a firing sequence $\sigma = \{t_1, t_2, \ldots, t_n\}$ such that M can be obtained from M_0 as firing of transitions $t_1, t_2, \ldots, t_n$.

A Petri Net structure PN = ⟨P, T, F, W⟩ without any specific initial marking is denoted simply by PN and Petri Net with the given initial marking is denoted by ⟨ PN, M_0⟩. For x∈ P (or T), *x and x* are the set of input transitions (or places) and the set of output transitions (or places), respectively. Further| *x | and | x* | stands for number of the input transitions (or places) and the output transitions (or places), respectively.

For a PT-Net, a path is a sequence of nodes ρ = ⟨ $x_1, x_2 \ldots x_n$ ⟩ where $(x_i, x_{i+1}) \in F$ for i = 1, 2… n-1. ρ is said to be elementary if and only if it does not contain the same node more than once and a cycle is a sequence of places ⟨ $p_1, p_2, \ldots p_n$ ⟩ such that there exist $t_1, t_2, \ldots, t_n \in T$: ⟨ $p_1, t_1, p_2, t_2, \ldots, p_n, t_n$ ⟩ forms an elementary path and $(t_n, p_1) \in F$ [30].

*Definition 2.2*For a PT-Net PN = ⟨P, T, F, W⟩, a transition t is said to be enabled at a marking M if and only if ∀ p∈ *t: M (p) > W (p, t). On firing t, M is changed to M' such that ∀ p∈ P: M'(p) = M (p) -W (p, t) + W (t, p).And is denoted as, M [PN, t⟩ M' or M [t⟩ M'.

Definition 2.3 For a PT-Net ⟨PN, M_0⟩, a sequence of transitions σ= ⟨$t_1, t_2 \ldots t_n$ ⟩is called a firing sequence if and only if M_0 [t_1⟩... [t_n⟩ M_n. In notation we use, M_0 [PN, σ ⟩ M_n or M_0 [σ⟩ M_n.

Definition 2.4 For a PT-Net ⟨PN, M_0⟩, a marking M is said to be reachable if and only if there exists a firing sequence a such that M_0 [ρ⟩ M. symbolically, M_0 [PN,*⟩ M or M_0 [*⟩ M. [PN, M_0⟩ or [M_0⟩ represents the set of all reachable markings of ⟨PN, M_0⟩.

Definition 2.5 A marked Petri net is an ordinary PT-Net PN = ⟨P, T, F, W⟩ such that ∀ p ∈ P: | *p | = | p* | = 1.

Liveness, boundedness, safeness, reversibility and conservativeness are well known properties of Petri nets. Liveness implies deadlock freeness. Boundedness refers to the property that the system is free from any potential

capacity overflow. Safeness and conservativeness are two special cases of boundedness. Reversibility refers to the capability of a system of being recovered or reinitialised from any reachable state. In general, liveness, boundedness and reversibility collectively characterise a robust or well-behaved system.

Definition 2.6. For a PT-net ⟨PN, M_0⟩, a transition t is said to be live if and only if ∀ M∈ [M_0⟩, there exists an M': M [*⟩ M' [t⟩. ⟨PN, M_0⟩ is said to be live if and only if every transition is live.

Definition 2.7. For a PT-net ⟨PN, M_0⟩, a place p is said to be k-bounded (or bounded) if and only if ∀M ∈ [M_0⟩ : M (p) < k, where k > 0. ⟨PN, M_0⟩ is said to be k-bounded (or bounded) if and only if every place is k-bounded.

Definition 2.8. A PT-net ⟨PN, M_0⟩, is said to be safe if and only if every place is 1-bounded.

Definition 2.9. A PT-net ⟨PN, M_0⟩, is said to be reversible if and only if ∀M∈ [M_0⟩: M [*⟩ M_0.

Definition 2.10 A PT-net ⟨PN, M_0⟩, is said to be well-behaved if and only if it is live, bounded and reversible.

Definition 2.11 Let PN = ⟨P, T, F, W⟩, be a PT-net, where P = {p_1, p_2,…, p_m} and T = {t_1, t_2,…, t_n}. The incidence matrix of N is an m × n matrix V whose typical entry v_i= W(pi,tj)- W(tj,pi) represents the change in number of tokens in p_i after firing t_j once, for i = 1, 2, ... , m and j = 1, 2, ... , n.

Definition 2.12 A PT-net PN = ⟨P, T, F, W⟩, is said to be conservative if and only if there exists an m-vector α> 0 such that αV = 0, where m = | P | and V is the incidence matrix of PN.

Fig. 3 shows a PT-net ⟨PN, M_0⟩ which is live, bounded, safe, reversible and conservative.

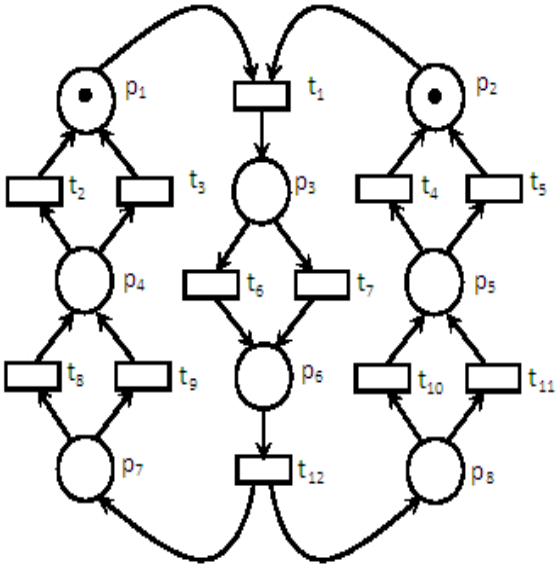

Fig.3 A live, bounded, safe, reversible and conservative Petri net.

III. AUGMENTED MARKED PETRI NET GRAPH AND ITS COMPOSITION

AMPN-structure refers to a augmented-marked-Petri net structure. In the literature, augmented-marked-Petri net is not well known, as compared to other sub-classes of Petri nets such as free-choice nets [33]. However, they possess many special properties pertaining to liveness, boundedness and reversibility. This section introduces Dynamic augmented marked Petri net and their special properties.

3.1 Augmented Marked Petri Net: An Augmented marked Petri net AMPN=⟨MPN, M_0; R⟩ is a Marked Petri net graph ⟨MPN, M_0 ⟩, with a specific subset of places R called resource places, Such that: (a) every place in R is marked by marking M_0.(b) An marked Petri net graph⟨MPN', M_0'⟩ can be obtained from ⟨MPN, M_0 (t); R⟩ by removing the places in R and their associated arcs. (c) For each place p∈R, there exist k > 1 pairs of transitions D_k = {(t_{s1}, t_{r1}), (t_{s2}, t_{r2}),..., (t_{sk}, t_{rk})} such that p* = { t_{s1}, t_{s2},…, t_{sk}} ∈ T and *p = {t_{r1},t_{r2},..., t_{rk} }∈ T and that, for each (t_{si}, t_{ri}) ∈ D_k, there exists an elementary path in AMPN say, ρ_{pi} connecting t_{si} to t_{ri}. (d) In ⟨MPN', M_0' (t)⟩, every cycle is marked and no ρ_{pi} is marked, see Fig. 4. Here we replace ⟨MPN, M_0; R⟩ simply by ⟨PN, M_0; R⟩ in our paper.

3.2 Composite Augmented Marked Petri nets: Let ⟨AMPN1, M_{01}; R1⟩ and ⟨AMPN2, M_{02}; R2⟩ be two Augmented marked Petri net graphs, in which R1' = { r_{11}, r_{12},…,r_{1k} } ∈ R1 and R2' = {r_{21}, r_{22} ,..., r_{2k}}∈ R2 are the common resource places. Let r_{11} and r_{21} be fused as one single place r_1, r_{12} and r_{22} into r_2, ..., r_{1k} and r_{2k} into r_k, then the resulting net is also an Augmented marked Petri net graph ⟨AMPN, M_0;R⟩, where R=(R1\R1') ∪ (R2\R2')∪{ r_1, r_2,…,r_k }.This net ⟨AMPN, M_0;R⟩ is called the composite Augmented Marked Petri Net graph of ⟨AMPN1, M_{01}; R1⟩ and ⟨AMPN2, M_{02}; R2⟩ via a set of common resource places { (r_{11},r_{21}), (r_{12}, r_{22}),…, (r_{1k}, r_{2k}). RF = { r_1, r_2,… r_k} is called the set of fused resource places that are obtained after fusing (r_{11},r_{21}), (r_{12}, r_{22}),…, (r_{1k}, r_{2k}).Fig.5(a) and 5(b) show two Augmented marked Petri net graphs ⟨AMPN1, M_{01}; R1⟩and⟨AMPN2, M_{02}; R2⟩, Fig. 5 shows the composite Augmented marked Petri net ⟨AMPN,M_0(t);R⟩of⟨AMPN1, M_{01}; R1⟩and⟨AMPN2, M_{02}; R2⟩ via (r_{11},r_{21}), where RF = { r_1}. A ⟨PN, M_0⟩ be a PT-net, where R = {r_1, r_2, ..., r_k} is the set of marked places such that | *r_i | > 0 and | r_i* | > 0 for i = 1, 2, k. ⟨PN, M_0⟩ is said to be of an AMPN-structure if and only if ⟨PN, M_0; R⟩ is an Augmented Marked Petri Net. For a PT-net ⟨PN, M_0⟩, a set of places S is called a siphon if and only if *S ⊂ S*. S is said to be minimal if and only if there does not exist any siphon S' in N such that S'⊂ S. S is said to be empty at a marking M ∈ [M_0⟩ if and only if S contains no places which are marked by M. And a set of places Q is called a trap if and only if Q*⊂ *Q. Q is said to be maximal if and only if there does not

exist any trap Q' in N such that Q ⊂ Q'. Q is said to be marked at a marking M∈ [M_0⟩ if and only if Q contains at least one place which is marked by M.

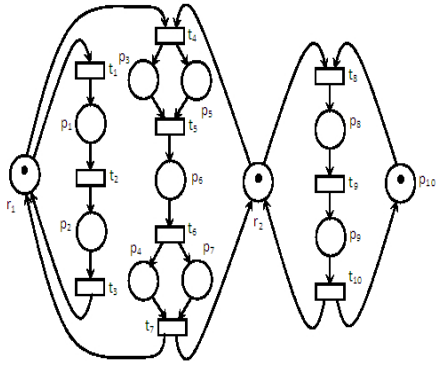

Fig.4 **Augmented Marked Petri net Graph**

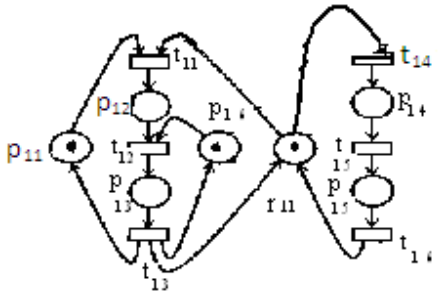

Fig. 5(a) (AMPN1, M01, R1)

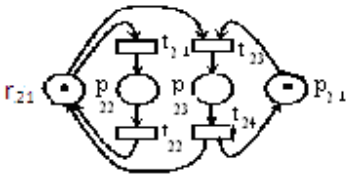

Fig. 5(B) (AMPN2, M02, R2)

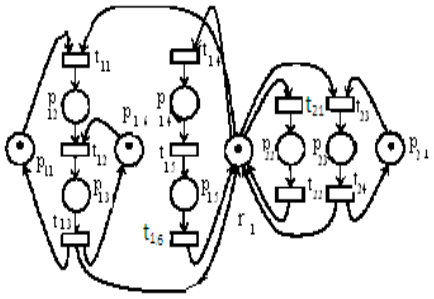

Fig. 5(c) Composition of two augmented Petri nets AMPN1 and AMPN2 in Fig. 5(a) and 5(b) via {r_{11}, r_{12}}

Property 3.1 [22]. An AMPN = ⟨PN, M_0; R⟩ is live and reversible if and only if it does not contain any potential deadlock. (Note: A potential deadlock is a siphon which would eventually become empty.)

For an AMPN = ⟨PN, M0; R⟩, a minimal siphon is called an R-siphon if and only if it contains at least one place in R.

Property 3.2 [22]. An AMPN = ⟨PN, M_0; R⟩ is live and reversible if and only if no R-siphons eventually become empty.

Property 3.3 [22]. An AMPN = ⟨PN, M_0; R ⟩is live and reversible if every R-siphon contains a marked trap.

For the AMPN = ⟨PN, M_0; R⟩ shown in Fig. 4, each R-siphon contains a marked trap, so

⟨PN, M_0; R⟩ is live and reversible.

Suppose an AMPN = ⟨PN, M_0; R⟩is transformed into a PT-net ⟨PN', M_0'⟩ as follows. For each p ∈R, k pairs of transitions {(t_{s1}, t_{r1}), (t_{s2}, t_{r2})… (t_{sk}, t_{rk}) }, replace with a set of places {q_1, q_2, q_k }, such that M_0'[q_i] = M_0[p] and q_i^* = { t_{si} } and $^*q_i$ = { t_{ri} } for i = 1, 2, ... , k. ⟨PN', M_0'⟩ is called the R-transform of ⟨PN, M_0; R⟩.

Property 3.4 [22]. Let ⟨PN', M_0'⟩ be the R-transform of an AMPN = ⟨PN, M_0; R⟩. AMPN is bounded and conservative if and only if every place in ⟨PN', M_0'⟩ belongs to a cycle.

Fig. 6 shows the R-transform ⟨PN', M_0'⟩ of the AMPN ⟨PN, M_0; R⟩ described in Fig.4. ⟨PN', M_0'⟩ is bounded, where every place belongs to a cycle. ⟨PN, M_0; R ⟩is bounded and conservative.

IV. Labelled Petri nets with an AMPN-Structure and Specification of object interaction scenarios as a Labelled Petri net

In this section, we show how an object interaction scenario can be formally specified as a Labelled Petri Net with an AMPN-structure (Step 1 of our proposed method).

A labelled Petri net is a Petri net, where labels are assigned to places and transitions. Usually, places are labelled by conditions to denote specific system substates in which the conditions hold and transitions are labelled by events (denote specific occurrences of the events).

Definition 4.1. A Labelled Petri net (LPN) is a 7-tuple *i.e.*, LPN = ⟨P, T, F, C, E, L_p, L_t⟩, where ⟨P, T, F ⟩ is an ordinary PT-net, C is a set of condition labels, E is a set of event labels, L_p: P → C is a function for assigning a condition label to every place, and L_t: T → E is a function for assigning an event label to every transition.

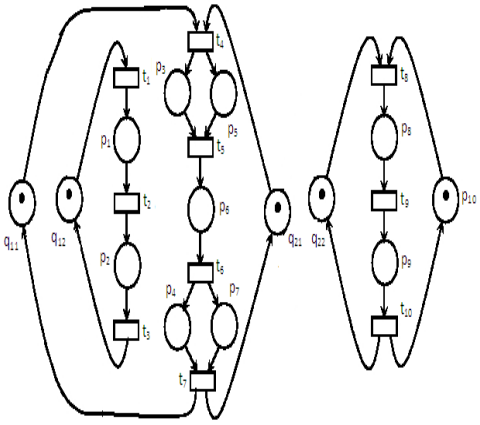

Fig.6 The R-transform of the augmented marked Petri net in Fig.4

Definition 4.2. Let LPN = ⟨P, T, F, C, E, L_p, L_t ⟩ be a Labelled Petri Net. A place p is said to be uniquely labelled in LPN if and only if ∀ p' ∈P: (L_p(p') = L_p(p)) ⇒ (p' = p). A transition t is said to be uniquely labelled in LPN if and only if ∀ t' ∈ T: (L_t(t') = L_t(t)) ⇒ (t' = t). LPN is said to be uniquely labelled if and only if all places and transitions are uniquely labelled.

Fig. 7 shows a typical labelled Petri net. Places p_3, p_4, p_5, p_6, p_9 and p_{10} are uniquely labelled, whereas p_1, p_2, p_7 and p_8 are not. As an illustration, condition label c_1 appears in p_1 and p_7, and c_2 in p_2 and p_8. Transitions t_3, t_4 and t_5 are uniquely labelled, where as t_1, t_2, t_6 and t_7 are not, as an example, event label e_1 appears in t_1 and t_6, and e_2 in t_2 and t_7. Therefore, the LPN is not uniquely labelled.

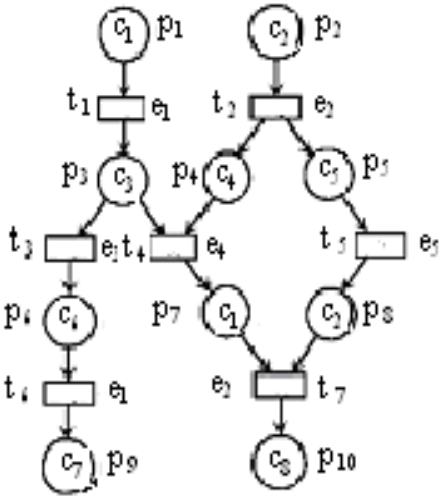

Fig. 7 A labelled Petri net which is not uniquely labeled

For an object interaction scenario specified as a Labelled Petri Net, the location where an event occurs is represented by a transition and the location of a condition by a place. The semantic meanings of conditions and events are denoted by the labels of the places and transitions respectively. For an event to occur, some conditions must be fulfilled in advance and some afterwards. These pre-conditions and post-conditions are represented by the pre-set and post-set of the transition representing the event.

Step 1 of the proposed method is to specify the given object interaction scenarios as Labelled Petri Nets with an AMPN-structure. Consider an object-oriented system involving two objects, x and y, of classes X and Y respectively. There are three typical interaction scenarios exhibited by x and y, specified as UML sequence diagrams and collaboration diagrams (BRJ99, RJB99) in Fig. 8. In conventional UML sequence diagrams and collaboration diagrams, there are no formal notations for denoting the pre-condition and post-condition of each event occurrence in an object interaction scenario. Therefore, for an explicit representation of the causal relationship between events and conditions, appropriate condition labels are appended to these diagrams.

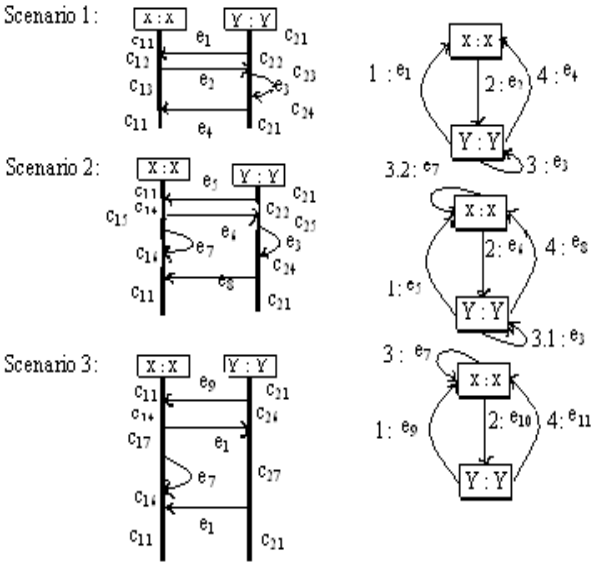

Fig 8. Object interaction scenarios in UML sequence diagrams and collaboration diagram

Fig. 9 shows object interaction Scenarios 1, 2 and 3, specified as LPNs ⟨LPN1, M_{01}⟩, ⟨LPN2, M_{02}⟩ and ⟨LPN3, M_{03}⟩ respectively. They all are of AMPN-structure.

⟨LPN1, M_{01} ⟩is constructed for representing scenario 1 as follows. For each location of a condition, a new place with a proper condition label is created. For example, p_{11} denotes a location of condition c_{11} for object x, so condition label $x.c_{11}$ is assigned to p_{11}. For each event occurrence, a new transition with a proper event label is constructed. For example, t_{11} denotes an occurrence of event e_1, so event label e_1 is assigned to t_{11}. The event occurrence has a pre-condition $x.c_{11}$ and a post-condition $x.c_{12}$. Hence, ${}^*t_{11} = \{p_{11}\}$ and $t_{11}^* = \{p_{12}\}$. Arcs between p_{11} (pre-condition) and t_{11} and between t_{11} and p_{12} (post-condition) are appended for denoting their causal relationships. The remaining location of conditions and events are created accordingly. Following the same rules, ⟨LPN2, M_{02}⟩ and ⟨LPN3, M_{03}⟩ are constructed for representing scenarios 2 and 3, respectively.

V. SYNTHESIZING AND ANALYZING THE INTEGRATED SYSTEM OF LABELLED PETRI NETS

After specifying the object interaction scenarios as AMPNs (Step 1 of the proposed method), we synthesize these scenarios into an integrated system. In principle, a scenario portrays partial system behaviors of how the objects are interacted in order to perform a specific functionality. These AMPNs are essentially partial system behavioural specifications which are to be synthesized together to form a single coherent whole. This section describes Step 2 of our proposed method - the synthesis of Labelled Petri Nets into an integrated net which represents the integrated system, and analysis of the system. The synthesis is based on the authors' earlier work

[30, 31]. It is made by fusing those places with refer to the same system initial state or condition. The integrated net so obtained is of AMPN-structure, For ⟨LPN, M_0; R), every R-siphon contains a marked place, and hence, would never become empty. According to Properties 3.2 and 3.3, ⟨LPN, M_0; R⟩ is live and reversible. Since every place in its R-transform is covered by cycles, according to Property 3.4,

⟨LPN, M_0; R⟩ is also bounded and conservative. Therefore, it can be concluded that the integrated system is well-behaved.

VI. ELIMINATING DUPLICATE LABELS FROM THE INTEGRATED NET

Consolidating the object interaction scenarios, the integrated net obtained from Step 2 of the proposed method serves to represent the system as a coherent integrated whole. In general, this integrated net is not necessarily uniquely labelled. For the integrated net ⟨LPN, M_0⟩ in Fig. 10 for example, places p_{15} and p_{26} have the same condition label $y.c_{22}$, and transitions t_{13} and t_{24} have the same event label e_3. This reflects the fact that the locations or conditions for occurrence of the same event may be different at different moments within a scenario or among different scenarios. Yet, every condition is eventually implemented as a unique system substate and every event as a unique operation. Thus the integrated net to be effectively used for implementation purposes, it needs to be uniquely labelled where all the duplicate condition labels and duplicate event labels are eliminated.

The elimination cannot be done by just fusing places with the same condition label, and transitions with the same event label. In this case the resulting net may exhibit firing sequences different from the original ones. In other words, the system behaviours may be distorted. Step 3 of the proposed method is to eliminate all duplicate labels while preserving the original firing sequences (event sequences). We describe this step in detail.

Definition 6.1. Let S be a uniquely labelled subnet of a Labelled Petri Net LPN. The pattern of S in LPN, denoted as Patt(LPN, S), is a condition-event net with an identical structure and label allocation S while ignoring the identities of places and transitions of S.

Let L_x and L_y be patterns of subnets in a Labelled Petri Net. $L_x \cup L_y$ and $L_x \cap L_y$ denote the union and intersection of L_x and L_y, respectively. $L_x \setminus L_y$ denotes the displacement of L_x from L_y. L_x and L_y are said to be disjoint if and only if $L_x \cap L_y = \emptyset$. For a Labelled Petri Net LPN, a uniquely labelled subnet S is called a common subnet if and only if there exists at least one uniquely labelled subnet S' such that $S' \neq S$ and Patt(LPN, S') = Patt(LPN, S). Let S be a pattern of the common subnets in LPN. [LPN, L] = {S | Patt(LPN, S) = L} represents the group of common subnets having the same pattern L. For a subnet S = ⟨P', T', F'⟩ of a PT-net, Pre(S) = (*P'\T') ∪ (*T'\P') is called the pre-set of S, Post(S) = (P'*\T') ∪ (T'*\P') is called the post-set of S, Head(S) = Pre(S)* ∩ (P' ∪ T') is called the head of S, and Tail(S) = *Post(S) ∩ (P' ∪ T') is called the tail of S.

Definition 6.2. A subnet S of a PT-net LPN = ⟨P, T, F⟩ is said to be of PP-type if and only if Head(S) ⊂ P and Tail(S) ⊂ P.

Fig. 11 shows a uniquely labelled subnet S which is PP-type. Fig. 12 shows the pattern of S.

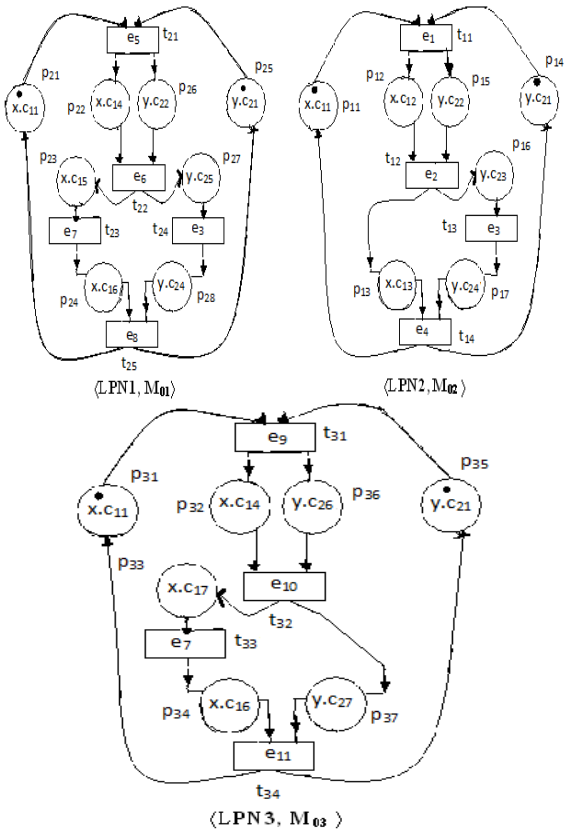

Fig.9 Labelled Petri nets representing the objects interaction scenarios in Fig.8

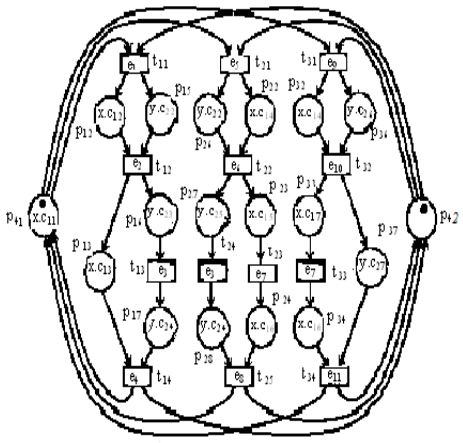

Fig. 10 The integrated net obtained by synthesizing the labelled Petri nets in Fig. 9.

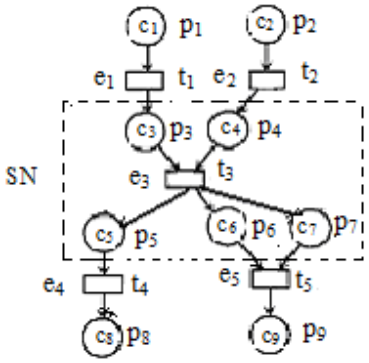

Fig. 11

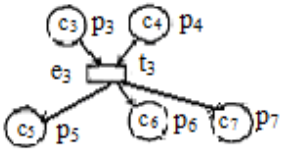

Fig.12

We propose to eliminate duplicate labels by fusion of common subnets, as outlined below.

Identify groups of common subnets for fusion. These groups of common subnets need to be maximal and disjoint for two reasons. First, the net obtained after the fusion will become uniquely labelled. Second, the number of groups of common subnets for fusion can be reduced to minimum as they are maximal.

Transformation of common subnets. For preservation of firing sequences, common subnets are transformed before fusion. Based on coloured Petri nets [36], a unique colour is assigned to each common subnet as colour labels of its ingoing and outgoing arcs. A token flowing into the common subnet is coloured according to the colour label of the ingoing arc. Its colour is reset as it flows out via the same colour-labelled outgoing arc. Besides, the subnets are converted to PP-type.

Fusion of transformed common subnets. The transformed common subnets of each group are fused into a single subnet. A uniquely Labelled Petri Net is ultimately obtained.

The following algorithm formally describes the elimination process. A detailed elaboration of the elimination process can be found in [35].

Algorithm for Elimination of Duplicate Labels from a Labelled Petri Net:

1. Identify maximal disjoint groups of common subnets:

1.1 Find all possible common subnets from LPN. Let $\tau = \{L_1, L_2, ..., L_n\}$ be their patterns.

1.2 Retain only the maximal patterns: Remove any L_i from τ if there exists $L_j \in \tau$ such that L_i is a sub-pattern of L_j and $\forall\ S_i \in [LPN, L_i], \exists\ S_j \in [LPN, L_j]$: S_i is a subnet of S_j.

1.3 Make the overlapping patterns disjoint : For every L_i, $L_j \in \tau$ such that $L_i \neq L_j$ and L_i and L_j are not disjoint, set $\tau = (\tau - \{ L_i, L_j\}) \cup \{L_i \cap L_j\} \cup \{ L_i \setminus L_j\} \cup \{ L_j \setminus L_i \}$.

1.4 Categorise the common subnets of LPN into groups $\{[LPN, L_i] \mid L_i \in \tau \}$.

2. For each group of common subnets $[LPN, L_i]$:

2.1 Convert each subnet $S \in [LPN, L_i]$ if S is not of PP-type:

2.1.1 For each transition $t_i \in$ Head(S): (a) Create dummy transition t_i' with unique label ε_i, dummy place p_i' with label φ_i, and arcs (t_i', p_i') and (p_i', t_i). (b) For each $p \in {}^*t_i$: Remove arc (p, t_i), and then create arc (p, t_i'). (c) Re-define S by including p_i' and (p_i', t_i).

2.1.2 For each transition $t_j \in$ Tail(S): (a) Create dummy transition t_j' with unique label ε_j, dummy place p_j' with label φ_j, and arcs (t_j, p_j') and (p_j', t_j'). (b) For each $p \in t_j^*$: Remove arc (t_j, p), and then create arc (t_j', p). (c) Re-define S by including p_j' and (t_j, p_j').

2.2 Assign a unique colour label K for each subnet $S \in [LPN, L_j]$:

2.2.1 For each arc (t_i, p_i) where $t_i \in$ Pre(S) and $p_i \in$ Head(S): Assign colour label K to (t_i, p_i).

2.2.2 For each arc (p_j, t_j) where $p_j \in$ Tail(S) and $t_j \in$ Post(S): Assign colour label K to (p_j, t_j).

2.3 Fuse the common subnets in $[LPN, L_i]$ into one single subnet.

We apply the algorithm for eliminating the duplicate labels for the integrated net (LPN, M_0) in Fig. 10. The obtained uniquely Labelled Petri Net (LPN', M_0') is given in Fig. 13.

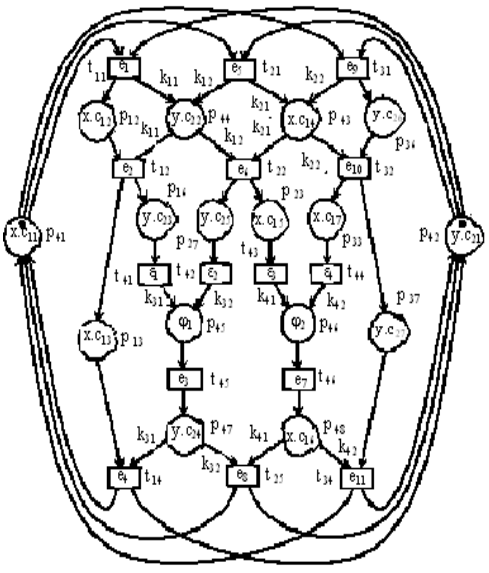

Fig.13 The uniquely labelled Petri net obtained after eliminating duplicate labels from the integrated net in Fig.10

VII. OBTAINING OBJECT-BASED BEHAVIOURAL SPECIFICATIONS

In this section, we implement Step 4 of our proposed method to obtain the individual object-based behavioural specifications. These individual object-based behavioural specifications are obtained by projecting the integrated net onto individual objects.

The projection is made by ignoring those places, transitions and arcs which are irrelevant to the object concerned. The projected net so obtained serves as the object behavioural specifications.

Consider the integrated net ⟨LPN', M_0'⟩ in Fig. 13. The projection onto object x is obtained as follows. We keep those places with object label x (including dummy places) and those transitions (including dummy transitions) having at least one input place or output place labelled by x, as well as their associated arcs. Similarly, for the projection onto object y, we keep those places with object label y (including dummy places) and those transitions (including dummy transitions) having at least one input place or output place labelled by y, as well as their associated arcs.

Fig. 14 shows the projections ⟨LPNx, M_0x⟩ and ⟨LPNy, M_0y⟩ obtained by projecting the net ⟨LPN', M_0' ⟩in Fig. 13 onto objects x and y, respectively. ⟨LPNx, M_0x ⟩ and ⟨LPNy, M_0y ⟩ are uniquely labelled, simply because ⟨LPN', M_0'⟩ is uniquely labelled. They serve as the behavioural specifications for objects x and y, where conditions and events are uniquely represented.

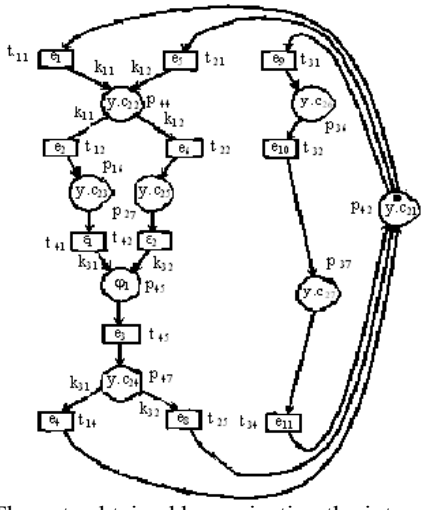

Fig. 14 The nets obtained by projecting the integrated net in Fig. 13 onto objects x and y.

VIII. LABELLED INTEGRATED NET OF OFFICE ACCESS CONTROL SYSTEM AND ITS PROJECTION ONTO INDIVIDUAL OBJECTS

An *Office Access Control System (OACS)* is a system used in an academic or a company for controlling staff access to its offices and laboratories. Among these offices and laboratories, some can be accessed by all staff members while other by authorised staff only during specified time periods. For controlling the staff access, every entrance is implemented with a card-reader, an emergency switch and an electronic lock, all being connected to a centralized server. The server maintains the access privileges and validates every access to the offices/laboratories. There are three typical cases for each access request.

Authorised access (U_j). If a staff member wants to access an office/laboratory, he/she has to present his/her staff card via a card-reader. Access is granted and the door is unlocked for thirty seconds and then re-locked.

Unauthorised access (U_2). A staff member wants to access an office/laboratory. He/She presents his/her staff card via a card-reader. If the Access is not granted then the door remains locked.

Emergency access (U_3). A staff member wants to access an office/laboratory for emergency. He/She presses the emergency key. The door is unlocked immediately, until it is reset by a security officer.

From the object-oriented perspectives, the server (s: Server) and doors (d: Door) are objects of the Office Access Control System. They are interacting with each other in order to perform the above system functionalities. For each of the access types U1, U2 and U3, there are three corresponding object interaction scenarios.

Fig. 15 shows these object interaction scenarios specified as labelled Petri nets of the Office Access Control System, where appropriate condition labels are appended for denoting the pre-conditions and post -conditions for each event occurrence.

Legends for condition labels and event labels:

C_{11}	Server is ready	e_1	Request for access is request
C_{12}	Server is processing access request	e_2	Access is granted
C_{13}	Server is waiting for re-lock	e_3	Time expires after access granted
C_{14}	Server is writing log(successful access)	e_4	Successful access is committed
C_{15}	Server is writing log(unsuccessful access)	e_5	Access is not granted
C_{16}	Server is waiting for emergency reset	e_6	Unsuccessful access is committed
C_{17}	Server is writing log(emergency access)	e_7	Request for emergency access is received
C_{21}	Door is locked	e_8	Door is reset to normal
C_{22}	Door is waiting for response	e_9	Emergency access is committed
C_{23}	Door is unlocked (Successful access)		
C_{24}	Door is unlocked (emergency access)		

Step 1 of the proposed method is to specify object interaction scenarios as Labelled Petri Nets. Fig. 15 shows the Labelled Petri Nets ⟨LPN1, M_{01}⟩, ⟨LPN2, M_{02}⟩ and ⟨LPN3, M_{03}⟩ representing the object interaction scenarios for U1, U2 and U3, respectively.

Step 2 of the proposed method is to synthesise the Labelled Petri Net into an integrated system, and analyse the system on its liveness, boundedness, reversibility and conservativeness. ⟨LPN1, M_{01}⟩, ⟨LPN2, M_{02}⟩ and ⟨LPN3, M_{03} ⟩are synthesised into an integrated net ⟨LPN, M_0⟩ by fusing those places which refer to the same system initial states or conditions : Places p_{11}, p_{21} and p_{31} are fused into one place p_{41}, and p_{15}, p_{24} and p_{34} into p_{42}. Fig. 16 shows the integrated net ⟨LPN, M_0 ⟩so obtained.

The integrated net ⟨LPN, M_0 ⟩is of a AMPN-structure. Let R = {p_{41}, p_{42}}. For ⟨LPN, M_0; R⟩, every R-siphon contains a marked place and hence would never become empty. According to Properties 3.2 and 3.3, ⟨LPN, M_0; R⟩ is live and reversible. Since every place in its R-transform is covered by cycles, according to Property 3.4, ⟨LPN, M_0; R⟩ is also bounded and conservative. Therefore, it may be concluded that the Office Access Control System is well-behaved.

As shown in Fig. 16, ⟨LPN, M_0⟩ is not uniquely labelled as it contains duplicate labels. For example, place p_{12} and p_{22} have the same condition label $s.c_{12}$ and transitions t_{11} and t_{21} have the same event label e_1. Since every condition is eventually implemented as a unique substate and every event as a unique operation, for an effective use of integrated net for implementation purposes, these duplicate labels must be eliminated.

We use Step 3 to eliminate duplicate condition labels and duplicate event labels from the integrated net ⟨LPN, M_0⟩ by fusing the common subnets. The elimination process is done by applying the algorithm described in Section 6. Fig. 17 shows the uniquely Labelled Petri Net ⟨LPN', M_0'⟩.

Step 4 of the proposed method is to obtain the individual object-based behavioural specification as projections of the integrated net onto the objects. The projection is made by ignoring those places, transitions and arcs which are irrelevant to the object concerned.

Consider the integrated net ⟨LPN', M_0'⟩ in Fig. 17. For the projection onto object s (the server object), we keep those places with object label s (including dummy places) and those transitions (including dummy transitions) having at least one input place or output place labelled by s, as well as their associated arcs. Similarly, for the projection onto object d (the door object), we keep those places with object label d (including dummy places) and those transitions (including dummy transitions) having at least one input place or output place labelled by d, as well as their associated arcs.

Fig. 18 shows the projections of ⟨LPNs, M_0s⟩ and ⟨LPNd, M_0d⟩ obtained by projecting the integrated net ⟨LPN', PM_0') in Fig. 17 onto objects s and d, respectively. As the integrated net ⟨LPN', M_0'⟩ is uniquely labelled, its projections ⟨LPNs, M_0s⟩ and ⟨LPNd, M_0d⟩ are also uniquely labelled, where every condition or event is uniquely represented. ⟨LPNs, M_0s⟩ and ⟨LPNd, M_0d⟩ then serve as the behavioural specifications for the server (s: Server) and door (d: Door) objects, respectively.

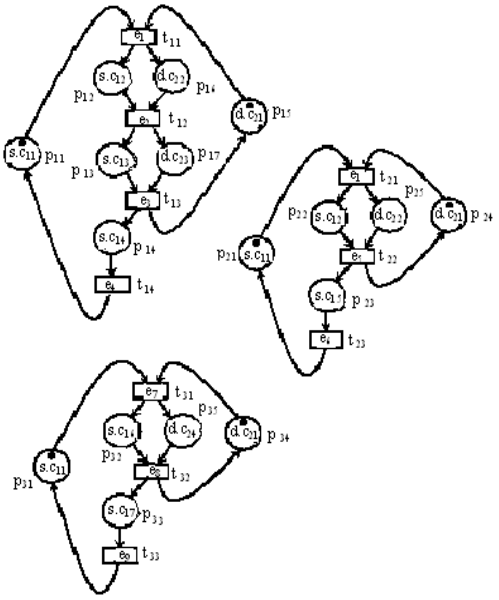

Fig. 15 labelled Petri nets representing the object interaction scenarios of U1, U2 and U3.

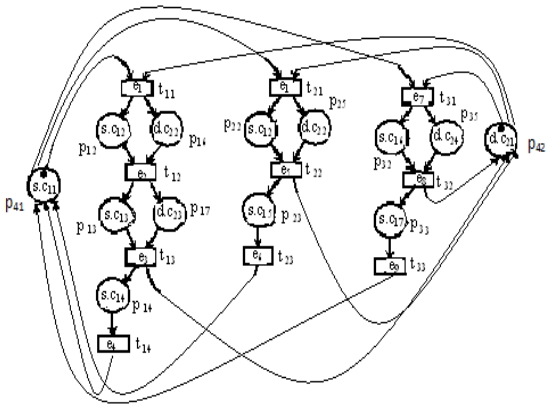

Fig. 16 The integrated net obtained by synthesizing the labelled Petri nets in Fig. 15

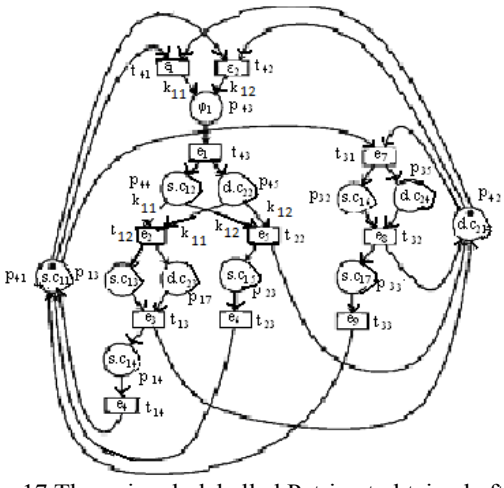

Fig. 17 The uniquely labelled Petri net obtained after eliminating duplicate labels from the integrated net in Fig.16

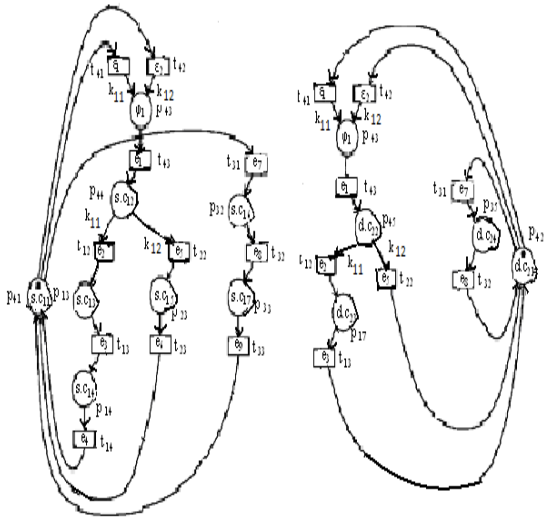

Fig. 18 The nets obtained by projecting the integrated net in Fig. 17 onto objects s and d.

CONCLUSION

One of the most difficult tasks in object-oriented system design is to ensure that the derived object-based behavioural specifications reflect exactly the given object interaction scenarios and that the system is well-behaved. In this paper, a Petri-net-based method is proposed for model the ACS. It begins with specifying each object interaction scenario as a labelled Petri net with an AMPN-structure. These labelled Petri nets are synthesized into a single integrated net which represents the integrated system. By making use of the special properties of the AMPN-structure, the system can be effectively analysed on its liveness, boundedness, reversibility and conservativeness. Duplicate labels are then eliminated by fusing common subnets, so as to attain a uniquely Labelled Petri Net on which individual object-based behavioural specifications are obtained as projections. The proposed method can be implemented as tool to support object-oriented system design. By capturing the functional requirements of a system as a set of object interaction scenarios, it helps in performing rigorous system synthesis and analysis. The correctness of this refinement can be assured. Moreover, the object-based behavioral specifications so obtained can be readily used for code generation. This inevitably contributes towards smooth transitions from functional requirements through design to implementation for object-oriented system development.

REFERENCES

[1]B.Baugarten,"*Petri Nets basics and application",* 2nd ed., Berlin: spectrum akademischer Verlag, (1996).

[2] B. Dano, H. Briand and F. Barbier, "*Progressing Towards Object-Oriented Requirements Specifications Using the Use Case Concept*", *Proceedings of the IEEE Symposium and Workshop on Engineering of Computer-Based Systems,* pp. 450-456, IEEE Computer Society Press (1996).

[3] B. Dano, H. Briand and F. Barbier, "*An Approach Based on the Concept of Use Case to Produce Dynamic Object-Oriented Specifications*", *Proceedings of the IEEE International Symposium on Requirements Engineering,* pp. 54-64, IEEE Computer Society Press (1997).

[4] C.A. Petri, "*Kommunikation mit Automaten.*" *Bonn: institute fur Instrumentelle Mathematik, Schriften des MM Nr. 3, 1962. Also, English translation,"Communication with Auto-mata*." New York: Griffiss Air Force Base.Tech. Rep.RADC-TR-65-377, vol.1, (1966).

[5] C.A. Petri, "*Fundamentals of a theory of asynchronous information flow*," in Proc. of IP Congress 62, pp. 386-390,(1963).

[6] D. Rosenberg ,"*Use Case Driven Object Modeling with UML : A Practical Approach",* Addison-Wesley(1999).

[7] D. Rosenberg and K. Scott, "*Applying Use Case Driven Object Modeling with UML",* Addison-Wesley (2001),.

[8] E. Smith , "*On Net Systems Generated by Process Foldings*", *Advances in Petri Nets, Lecture Notes in Computer Science,* Vol. 524, pp. 253-295, Springer-Verlag (1991).

[9] F. Commoner, A. W. Holt, S. Even and A. Pnueli, "*Marked directed graph"*, Journal of Computer and System Sciences- pp. 511-523, (1971*)*.

[10] F. Bordeleau and R.J.A. Buhr, "*UCM-ROOM Modelling : From Use Case Maps to Communicating State Machines*", *Proceedings of the IEEE International Symposium and Workshop on Engineering of Computer-Based Systems,* pp. 169-178, IEEE Computer Society Press (1997),.

[11] F. Bordeleau, J.P. Corriveau and B. Selic, "*A Scenario-Based Approach to Hierarchical State Machine Design*", *Proceedings of the International Symposium on Object-Oriented Real-Time Distributed Computing,* pp. 78-85, IEEE Computer Society Press(2000).

[12] G. Schneider and J.P. Winters, "*Applying Use Cases",* Addison-Wesley (1998).

[13] G. Booch, J. Rumbaugh and I. Jacobson, " *The Unified Modeling Language : User Guide",* Addison-Wesley (1999).

[14] I. Graham, "*Object-Oriented Methods :Principles and Practice",* Addison-Wesley (2001).

[15] I. Jacobson , "*Object-Oriented Software Engineering : A Use-Case-Driven Approach",* Addison-Wesley (1992).

[16] I. Jacobson, G. Booch and J. Rumbaugh, "*The Unified Software Development Process",* Addition Wesley (1999).

[17] P. Kruchten , "*The Rational Unified Process :An Introduction",* Addison-Wesley (1999).

[18] J. Iivari, "Object Orientation as Structural, Functional and Behavioural Modelling: A Comparison of Six Methods for Object-Oriented Analysis", *Information and Software Technology,* Vol. 37, No. 3, pp. 155-163 (1995).

[19] J. Rumbaugh, I. Jacobson and G. Booch. "*The Unified Modeling Language: Reference Manua"l,* Addison-Wesley (1999).

[20] J. Arlow and I. Neustadt, "*UML and the Unified Process : Practical Object-Oriented Analysis and Design",* Addison-Wesley(2002).

[21] J. Desel and J. Esparza, "*Free-choice Petri Nets",* Cambridge University Press (1995).

[22] F. Chu and X. Xie,"*Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming*", *IEEE Transactions on Robotics and Automation,* Vol. 13, No. 6, pp. 793-804 (1997).

[23] J. Desel and W. Reisig "*Place Transition Petri Nets*", *Lectures on Petri Nets 1 : Basic Models, Lecture Notes in Computer Science,* Vol. 1491, pp.122-173, Springer-Verlag (1998).

[24] J. L., Peterson, "*Petri net Theory and the Modeling of Systems"*, Prentice Hall, INC, Englewood, Cliffs, and New Jercy, (1981).

[25] K. O. Chow and S. Yeung, "*Behavioural Modellingin Object-Oriented Methodologies*", *Information and Software Technology,* Vol. 38, No. 01, pp. 657-666 (1996).

B[26] K. Hiraishi, "*Construction of a Class of Safe Petri Nets by Presenting Firing Sequences*", *Application and Theory of Petri Nets, Lecture Notes in Computer Science,* Vol. 616, pp. 244-262, Springer-Verlag (1992).

[27] M. Glinz, "*A Lightweight Approach to Consistency of Scenarios and Class Models*", *Proceedings of the IEEE International*

Conference on Requirements Engineering, pp. 49-58, IEEE Computer Society Press(2000).

[28] P. Graubmann, "*The Construction of EN Systems from a Given Trace Behaviour*", *Advances in Petri Nets, Lecture Notes in Computer Science,* Vol. 340, pp. 133-153, Springer-Verlag(1988).

[29] S. Kirani and W.T. Tsai, "*Method Sequence Specification and Verification of Classes*", *Journal of Object-Oriented Programming,* Vol. 7, No. 6, pp. 28-38(1994).

[30] Sunita Kumawat and G.N. Purohit, "*Travelling Salesman's Problem in weighted directed graph: A Petri net Approach"* Proceedings of APORS 2009, pp.42, (2009).

[31] Sunita Kumawat and G.N. Purohit "*Synthesis of The Distributed Wireless Sensor Networks Base Nodes to a Common Sharing Server using Dynamic Augmented Marked Petri Net*" International Journal of Advanced Engineering Application, June issue (2010).

[32] T. Murata, "Petri Nets : Properties, Analysis and Applications", *Proceedings of the IEEE,* Vol.77, No. 4, pp. 541-580 (1989).

[33] W.J. Lee, S.D. Cha and Y.R. Kwon, "*Integration and Analysis of Use Cases Using Modular Petri Nets in Requirement Engineering*", *IEEE Transactions on Software Engineering,* Vol. 24, No. 12, pp. 1115-1103(1998).

[34] W. Reisig, "*Petri Nets and introduction* "Heidelberg: springer-verlag, (1985).

[35] K.S. Cheung and K.O. Chow, "Elimination of Duplicate Labels in Petri-Net-Based System Specification", *Proceedings of the International Conference on Computer and Information Technology,* pp. 932-936, IEEE Computer Society Press (2006),.

[36] K. Jensen, "Coloured Petri Nets", *Petri Nets : Central Models and Their Properties, Lecture Notes in Computer Science,* Vol. 254, pp. 248-299, Springer-Verlag (1986).

[37] Sunita Kumawat and G.N. Purohit,"*Travelling Salesman's Problem: A Petri net Approach"*International journal of computer and network security(IJCNS), vol.II, pp.19-24.(2010).

[38] Sunita Kumawat and G.N. Purohit, " *Extraction of non-Hamiltonian weighted directed graphs from a Hamiltonian weighted directed graph"* International Journal of Mathematical Sciences and Engineering Application(IJMSEA), vol. III.. (2010)

.