A Petri-Net Approach to Refining Object Behavioural Specifications

King-Sing Cheung
University of Hong Kong
Pokfulam, Hong Kong
E-mail : ks.cheung@hku.hk

Paul Kai-On Chow
City University of Hong Kong
Tat Chee Avenue, Hong Kong
E-mail : cspchow@cityu.edu.hk

Keywords: Petri net, object-oriented system, object-oriented design, behavioural specification

Received: April 12, 2008

In object-oriented system design, functional requirements are given and expressed as object interaction scenarios whereas implementation is based on classes of objects. One need to derive, from the given object interaction scenarios, object-based behavioural specifications which reflect exactly these object interaction scenarios for implementation purposes. In this paper, a Petri-net-based method is proposed for the refinement. It begins with specifying each object interaction scenario as a labelled Petri net with an AMG-structure. These labelled Petri nets are synthesised into a single integrated net which represents the integrated system. By making use of the special properties of the AMG-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 net on which individual object-based behavioural specifications are obtained as projections.

Povzetek: Uporabljen je pristop Petrijevih mrež za objektne specifikacije.

1 Introduction

In the past two decades, object orientation has been an influential discipline in software engineering[1]. 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) as shown in Figure 1 [1, 2, 3, 4, 5, 6, 7, 8, 9]. Structurally, objects with the same attributes are grouped into classes while classes having common attributes are generalised to form an inheritance hierarchy. Objects exhibit different behaviours on interacting with others, thus demonstrating different object interaction scenarios. This paper investigates the behavioural aspect of objects.

In object-oriented system design, the functional requirements of a system are given by the end-users as use cases - the typical cases of how a system can be used [10, 11]. These use cases are elaborated and expressed in terms of object interaction scenarios and specified as UML sequence diagrams and collaboration diagrams [11, 12, 13, 14, 15, 16]. We need to create, from the object interaction scenarios, object-based specifications delineating the behaviours of individual objects for detailed system design and implementation.

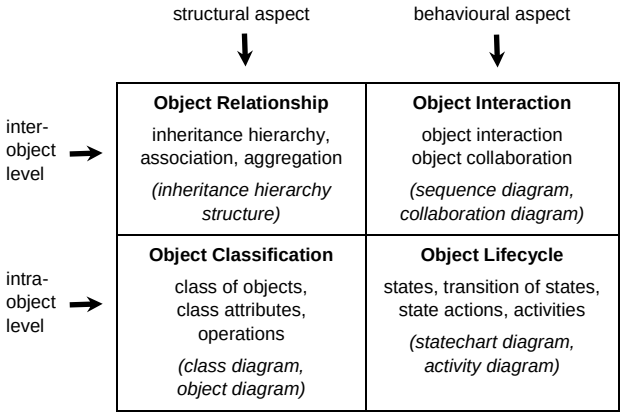


Figure 1: An object-oriented system by aspects and abstraction levels.

In this refinement process, at least the following problems have to be tackled.

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.

[1]This paper is an extended version of the authors' paper presented at the REFINE 2006 workshop.

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 [4, 17, 18, 19, 20]. 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 analyse whether the outcome system design is live, bounded and reversible.

In the literature, there are only a few approaches or methods for deriving an object-based behavioural specifications from a given set of use cases or object interaction scenarios. Bordeleau proposed an approach which takes a traceable progression from use cases to the object-based state machines [21, 22]. Dano proposed an approach where the use cases are synthesised into a system design according to some mapping rules [23, 24]. However, these approaches solve only trivial issues. The system design cannot be rigorously analysed on 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 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 [25]. 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 [26]. Hiraishi proposed a method for constructing a Petri net from a set of firing sequences [27]. 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 [28]. However, without concepts of object-orientation, these approaches and methods cannot be applied to object-oriented system design.

In this paper, based on Petri nets, we propose a method for refining 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 :

Step 1. Each object interaction scenario is specified as a labelled Petri net (labelled net) with an AMG-structure (i.e. structurally an augmented marked graph).

Step 2. The labelled nets are synthesised into an integrated net which serves to represent the system. Based on the properties of AMG-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.

Figure 2 shows an overview of the proposed method.

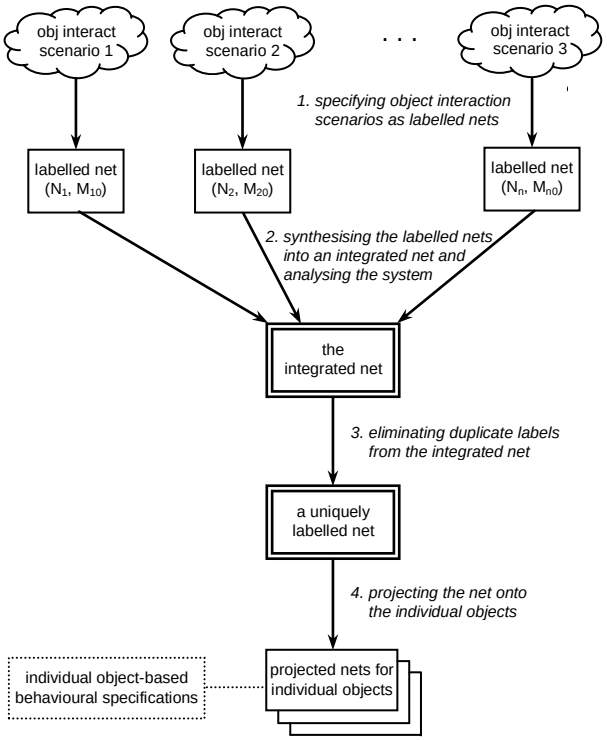


Figure 2: Overview of the proposed method.

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 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 an AMG-structure. By making use of the special properties of AMG-structure, the system can be effectively analysed 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 organised as follows. Section 2 provides the preliminaries to be used in this paper. Section 3 introduces the AMG-structure, where augmented marked graphs and their properties are described. In Section 4, we show the formal specification of object interaction scenarios as labelled nets (Step 1 of the proposed method). In Section 5, we focus on synthesising the labelled nets into an integrated system, and analysing the system properties (Step 2 of the proposed method). Section 6 then presents an algorithm for eliminating duplicate labels from the integrated net (Step 3 of the proposed method). In Section 7, we show how individual object-based behavioural specifications are obtained as projections of the integrated net (Step 4 of the proposed method). Section 8 gives a real-life example for illustration. Section 9 concludes this paper.

It should be noted that this paper primarily focus on refinement of object-based behavioural specifications - deriving individual object-based specifications from the object interaction scenarios. The structural aspect of an object-oriented system will not be investigated.

2 Preliminaries

This section provides the preliminaries for readers who are not familiar with Petri nets [29, 30, 31, 32].

A place-transition net (PT-net) is a directed graph consisting of two sorts of nodes called places and transitions, such that no arcs connect two nodes of the same sort. Graphically, a place is denoted by a circle, a transition by a box, and an arc by a directed line. A Petri net is a PT-net with some tokens assigned to its places, and the token distribution over its places is denoted by a marking function.

Definition 2.1. A place-transition net (PT-net) is a 4-tuple $N = \langle P, T, F, W \rangle$, where P is a set of places, T is a set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation and $W : F \rightarrow \{ 1, 2, ... \}$ is a weight function. N is said to be ordinary if and only if the range of W is { 1 }. An ordinary PT-net is usually written as $\langle P, T, F \rangle$.

Definition 2.2. Let $N = \langle P, T, F, W \rangle$ be a PT-net. For $x \in (P \cup T)$, ${}^\bullet x = \{ y \mid (y, x) \in F \}$ and $x^\bullet = \{ y \mid (x, y) \in F \}$ are called the pre-set and post-set of x, respectively. For $X = \{ x_1, x_2, \ldots, x_n \} \subseteq (P \cup T)$, ${}^\bullet X = {}^\bullet x_1 \cup {}^\bullet x_2 \cup \ldots \cup {}^\bullet x_n$ and $X^\bullet = x_1^\bullet \cup x_2^\bullet \cup \ldots \cup x_n^\bullet$ are called the pre-set and post-set of X, respectively.

Definition 2.3. For a PT-net $N = \langle P, T, F, W \rangle$, a path is a sequence of nodes $\langle x_1, x_2, ..., x_n \rangle$, where $(x_i, x_{i+1}) \in F$ for i = 1, 2, ..., n-1. A path is said to be elementary if and only if it does not contain the same node more than once.

Definition 2.4. For a PT-net $N = \langle P, T, F, W \rangle$, a cycle is a sequence of places $\langle p_1, p_2, ..., p_n \rangle$ such that $\exists\ t_1, t_2, ..., t_n \in T : \langle p_1, t_1, p_2, t_2, ..., p_n, t_n \rangle$ forms an elementary path and $(t_n, p_1) \in F$.

Definition 2.5. For a PT-net $N = \langle P, T, F, W \rangle$, a marking is a function $M : P \rightarrow \{ 0, 1, 2, ... \}$ where M(p) is the number of tokens in p. (N, M_0) represents N with an initial marking M_0.

Definition 2.6. For a PT-net $N = \langle P, T, F, W \rangle$, a transition t is said to be enabled at a marking M if and only if $\forall\ p \in {}^\bullet t : M(p) \geq W(p,t)$. On firing t, M is changed to M' such that $\forall\ p \in P : M'(p) = M(p) - W(p,t) + W(t,p)$. In notation, $M\ [N,t\rangle\ M'$ or $M\ [t\rangle\ M'$.

Definition 2.7. For a PT-net (N, M_0), a sequence of transitions $\sigma = \langle t_1, t_2, ..., t_n \rangle$ is called a firing sequence if and only if $M_0\ [t_1\rangle ... [t_n\rangle\ M_n$. In notation, $M_0\ [N,\sigma\rangle\ M_n$ or $M_0\ [\sigma\rangle\ M_n$.

Definition 2.8. For a PT-net (N, M_0), a marking M is said to be reachable if and only if there exists a firing sequence σ such that $M_0\ [\sigma\rangle\ M$. In notation, $M_0\ [N,*\rangle\ M$ or $M_0\ [*\rangle\ M$. $[N, M_0\rangle$ or $[M_0\rangle$ represents the set of all reachable markings of (N, M_0).

Definition 2.9. Let $N = \langle P, T, F, W \rangle$ 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 $m \times n$ matrix V whose typical entry $v_{ij} = W(p_i,t_j) - W(t_j,p_i)$ 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.

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 2.10. For a PT-net (N, M_0), a transition t is said to be live if and only if $\forall\ M \in [M_0\rangle, \exists\ M' : M\ [*\rangle\ M'\ [t\rangle$. (N, M_0) is said to be live if and only if every transition is live.

Definition 2.11. For a PT-net (N, M_0), a place p is said to be k-bounded (or bounded) if and only if $\forall\ M \in [M_0\rangle : M(p) \leq k$, where $k > 0$. (N, M_0) is said to be k-bounded (or bounded) if and only if every place is k-bounded.

Definition 2.12. A PT-net (N, M_0) is said to be safe if and only if every place is 1-bounded.

Definition 2.13. A PT-net (N, M_0) is said to be reversible if and only if $\forall\ M \in [M_0\rangle : M\ [*\rangle\ M_0$.

Definition 2.14. A PT-net (N, M_0) is said to be well-behaved if and only if it is live, bounded and reversible.

Definition 2.15. A PT-net $N = \langle P, T, F, W \rangle$ is said to be conservative if and only if there exists a m-vector $\alpha > 0$ such that $\alpha V = 0$, where $m = | P |$ and V is the incidence matrix of N.

Figure 3 shows a PT-net (N, M_0) which is live, bounded, safe, reversible and conservative.

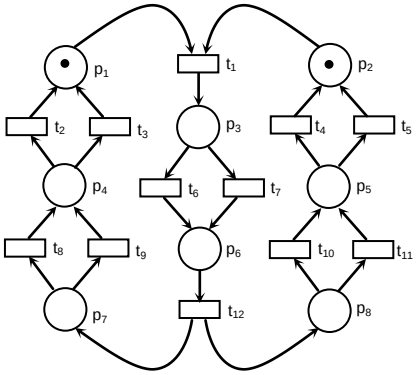


Figure 3. A live, bounded, safe, reversible and conservative PT-net.

3 AMG-structure and its properties

AMG-structure refers to an augmented-marked-graph structure. In the literature, augmented marked graphs are 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 augmented marked graphs and their special properties.

Definition 3.1 [34]**.** An augmented marked graph (N, M_0; R) is an ordinary PT-net (N, M_0) with a specific subset of places R, satisfying that : (a) Every place in R is marked by M_0. (b) The net (N', M_0') obtained from (N, M_0; R) by removing the places in R and their associated arcs is a marked graph. (c) For each place $r \in R$, there exist $k_r \geq 1$ pairs of transitions $D_r = \{ \langle t_{s1}, t_{h1} \rangle, \langle t_{s2}, t_{h2} \rangle, ..., \langle t_{skr}, t_{hkr} \rangle \}$, such that $r^\bullet = \{ t_{s1}, t_{s2}, ..., t_{skr} \} \subseteq T$ and $^\bullet r = \{ t_{h1}, t_{h2}, ..., t_{hkr} \} \subseteq T$ and that, for each $\langle t_{si}, t_{hi} \rangle \in D_r$, there exists in N' an elementary path ρ_{ri} connecting t_{si} to t_{hi}. (d) In (N', M_0'), every cycle is marked and no ρ_{ri} is marked.

Figure 4 shows an augmented marked graph (N, M_0; R), where $R = \{ r_1, r_2 \}$.

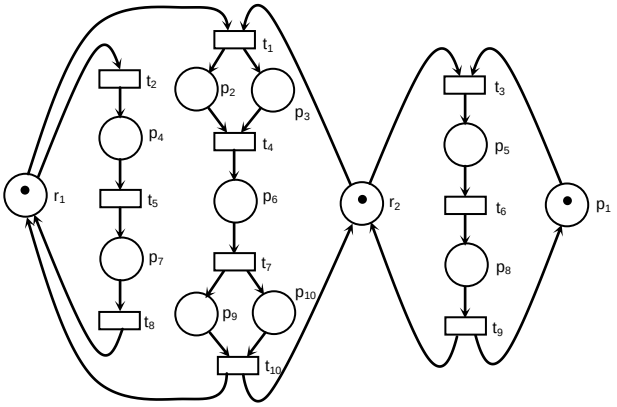


Figure 4. An augmented marked graph.

Definition 3.2. Let (N, M_0) be a PT-net, where $R = \{ r_1, r_2, \ldots, r_k \}$ is the set of marked places such that $| {}^\bullet r_i | > 0$ and $| r_i^\bullet | > 0$ for i = 1, 2, …, k. (N, M_0) is said to be of an AMG-structure if and only if (N, M_0; R) is an augmented marked graph.

Definition 3.3. For a PT-net (N, M_0), a set of places S is called a siphon if and only if ${}^\bullet S \subseteq S^\bullet$. S is said to be minimal if and only if there does not exist any siphon S' in N such that $S' \subset S$. S is said to be empty at a marking $M \in [M_0\rangle$ if and only if S contains no places which are marked by M.

Definition 3.4. For a PT-net (N, M_0), a set of places Q is called a trap if and only if $Q^\bullet \subseteq {}^\bullet Q$. Q is said to be maximal if and only if there does not exist any trap Q' in N such that $Q \subset Q'$. Q is said to be marked at a marking $M \in [M_0\rangle$ if and only if Q contains at least one place which is marked by M.

Property 3.1 [34]**.** An augmented marked graph 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.)

Definition 3.5. For an augmented marked graph (N, M_0; R), a minimal siphon is called an R-siphon if and only if it contains at least one place in R.

Property 3.2 [35, 36]**.** An augmented marked graph (N, M_0; R) is live and reversible if and only if no R-siphons eventually become empty.

Property 3.3 [34, 35, 36]**.** An augmented marked graph (N, M_0; R) is live and reversible if every R-siphon contains a marked trap.

For the augmented marked graph (N, M_0; R) shown in Figure 4, each R-siphon contains a marked trap. (N, M_0; R) is live and reversible.

Definition 3.6 [37]**.** Suppose an augmented marked graph (N, M_0; R) is transformed into a PT-net (N', M_0') as follows. For each $r \in R$, where $D_r = \{ \langle t_{s1}, t_{h1} \rangle, \langle t_{s2}, t_{h2} \rangle, ..., \langle t_{skr}, t_{hkr} \rangle \}$, replace r with a set of places $\{ q_1, q_2, ..., q_{kr} \}$, such that $M_0'[q_i] = M_0[r]$ and $q_i^\bullet = \{ t_{si} \}$ and ${}^\bullet q_i = \{ t_{hi} \}$ for i = 1, 2, ..., k_r. (N', M_0') is called the R-transform of (N, M_0; R).

Property 3.4 [37]**.** Let (N', M_0') be the R-transform of an augmented marked graph (N, M_0; R). (N, M_0; R) is bounded and conservative if and only if every place in (N', M_0') belongs to a cycle.

Figure 5 shows the R-transform (N', M_0') of the augmented marked graph (N, M_0; R) in Figure 4. (N', M_0') is bounded, where every place belongs to a cycle. (N, M_0; R) is bounded and conservative.

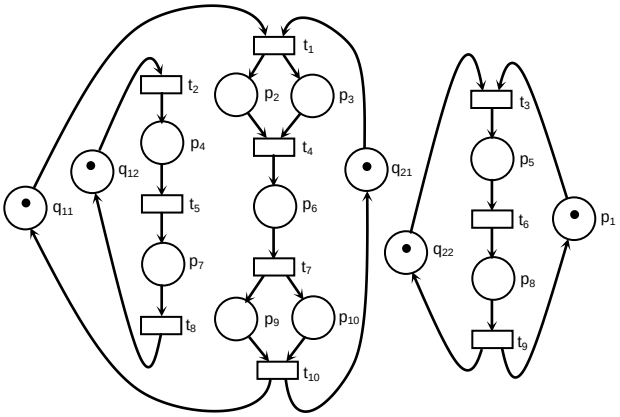


Figure 5. The R-transform of the augmented marked graph in Figure 4.

4 Specifying object interaction scenarios as labelled nets

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

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

Definition 4.1. A labelled Petri net (or labelled net) is a 7-tuple $N = \langle P, T, F, C, E, Lp, Lt \rangle$, where $\langle P, T, F \rangle$ is an ordinary PT-net, C is a set of condition labels, E is a set of event labels, $Lp : P \rightarrow C$ is a function for assigning a condition label to every place, and $Lt : T \rightarrow E$ is a function for assigning an event label to every transition.

Definition 4.2. Let $N = \langle P, T, F, C, E, Lp, Lt \rangle$ be a labelled net. A place p is said to be uniquely labelled in N if and only if $\forall p' \in P : (Lp(p') = Lp(p)) \Rightarrow (p' = p)$. A transition t is said to be uniquely labelled in N if and only if $\forall t' \in T : (Lt(t') = Lt(t)) \Rightarrow (t' = t)$. N is said to be uniquely labelled if and only if all places and transitions are uniquely labelled.

Figure 6 shows a typical labelled 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 for example, 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, whereas t_1, t_2, t_6 and t_7 are not, as for example, event label e_1 appears in t_1 and t_6, and e_2 in t_2 and t_7. Therefore, the labelled net is not uniquely labelled.

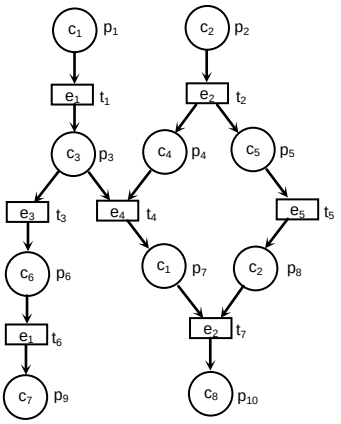


Figure 6. A labelled net which is not uniquely labelled.

For an object interaction scenario specified as a labelled 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 nets with an AMG-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 Figure 7. 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.

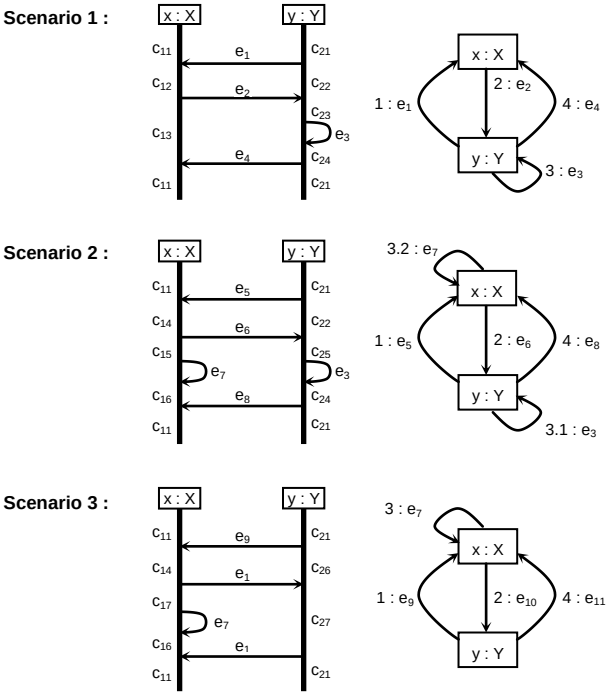


Figure 7. Object interaction scenarios in UML sequence diagrams and collaboration diagram.

Figure 8 shows object interaction Scenarios 1, 2 and 3, specified as labelled nets (N_1, M_{10}), (N_2, M_{20}) and (N_3, M_{30}) respectively. They all are of AMG-structure.

(N_1, M_{10}) 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, ${}^\bullet t_{11} = \{ p_{11} \}$ and $t_{11}{}^\bullet = \{ p_{12} \}$. 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 rest locations of conditions and events are created accordingly. Following the same rules, (N_2, M_{20}) and (N_3, M_{30}) are constructed for representing scenarios 2 and 3, respectively.

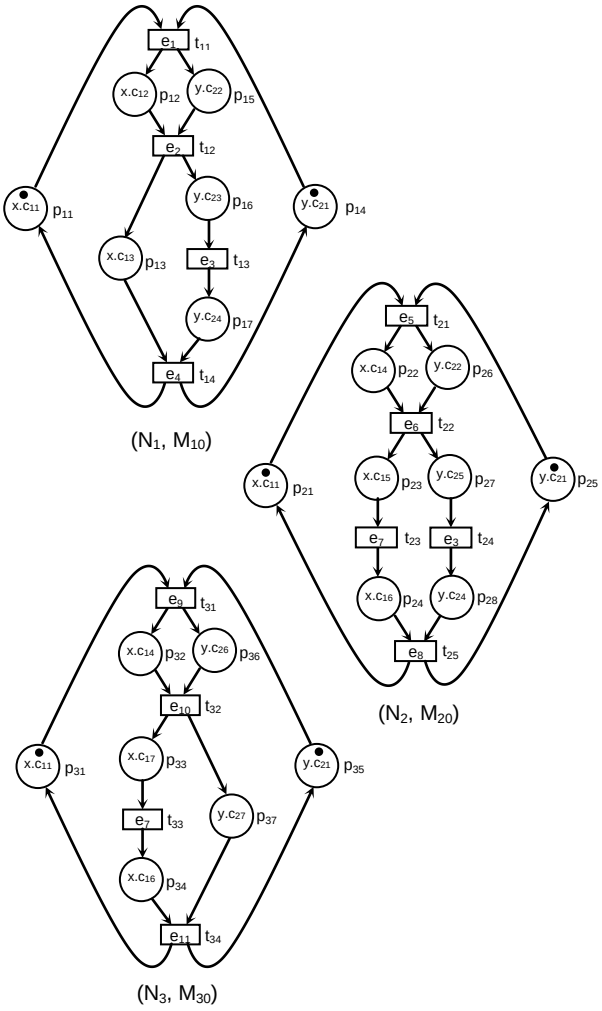


Figure 8. Labelled nets representing the object interaction scenarios in Figure 7.

5 Synthesising and analysing the integrated system

After specifying the object interaction scenarios as augmented marked graphs (Step 1 of the proposed method), we synthesise these scenarios into an integrated system. In principle, a scenario portrays partial system behaviours of how the objects are interacted in order to perform a specific functionality. These augmented marked graphs are essentially partial system behavioural specifications which are to be synthesised together to form a single coherent whole.

This section describes Step 2 of our proposed method - the synthesis of labelled nets into an integrated net which represents the integrated system, and analysis of the system. The synthesis is based on the authors' earlier work on use-case-driven system design [38]. It is made by fusing those places with refer to the same system initial state or condition. The integrated net so obtained is of AMG-structure, so its liveness, boundedness, reversibility and conservativeness can be effectively analysed by making use of the special properties of augmented marked graphs.

Consider the labelled nets (N_1, M_{10}), (N_2, M_{20}) and (N_3, M_{30}) in Figure 8. Places p_{11} in (N_1, M_{10}), p_{21} in (N_2, M_{20}) and p_{31} in (N_3, M_{30}) refer to the same condition $x.c_{11}$. Also, places p_{15} in (N_1, M_{10}), p_{24} in (N_2, M_{20}) and p_{34} in (N_3, M_{30}) refer to the same condition $y.c_{21}$. Hence, 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}.

Figure 9 then shows the integrated net (N, M_0) obtained after synthesising (N_1, M_{10}), (N_2, M_{20}) and (N_3, M_{30}). (N, M_0) is of an AMG-structure, meaning that it is structurally an augmented marked graph $(N, M_0; R)$, where $R = \{ p_{41}, p_{42} \}$.

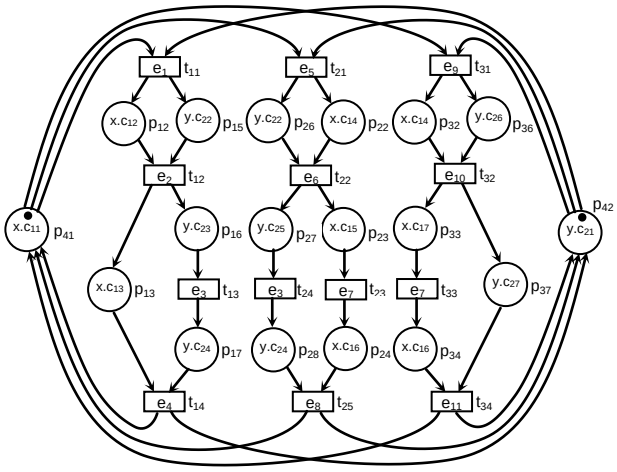


Figure 9. The integrated net obtained by synthesising the labelled nets in Figure 8.

For $(N, M_0; R)$, every R-siphon contains a marked place, and hence, would never become empty. According to Properties 3.2 and 3.3, $(N, M_0; R)$ is live and reversible. Since every place in its R-transform is covered by cycles, according to Property 3.4, $(N, M_0; R)$ is also bounded and conservative. Therefore, it can be concluded that the integrated system is well-behaved.

6 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 (N, M_0) in Figure 9 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. Therefore, in order for the integrated net to be effectively used for implementation purposes, it need 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. This is because 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). This section describes this step in details.

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

Definition 6.2. Let L_x and L_y be patterns of subnets in a labelled 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 $L_x \cap L_y = \varnothing$.

Definition 6.3. For a labelled net N, 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(N, S') = Patt(N, S). Let S be a pattern of the common subnets in N. [N, L] = { S | Patt(N, S) = L } represents the group of common subnets having the same pattern L.

Definition 6.4. For a subnet $S = \langle P', T', F' \rangle$ of a PT-net, $Pre(S) = ({}^\bullet P' \setminus T') \cup ({}^\bullet T' \setminus P')$ is called the pre-set of S, $Post(S) = (P'^\bullet \setminus T') \cup (T'^\bullet \setminus P')$ is called the post-set of S, $Head(S) = Pre(S)^\bullet \cap (P' \cup T')$ is called the head of S, and $Tail(S) = {}^\bullet Post(S) \cap (P' \cup T')$ is called the tail of S.

Definition 6.5. A subnet S of a PT-net $N = \langle P, T, F \rangle$ is said to be of PP-type if and only if $Head(S) \subseteq P$ and $Tail(S) \subseteq P$.

Figure 10 shows a uniquely labelled subnet S which is PP-type. Figure 11 shows the pattern of S.

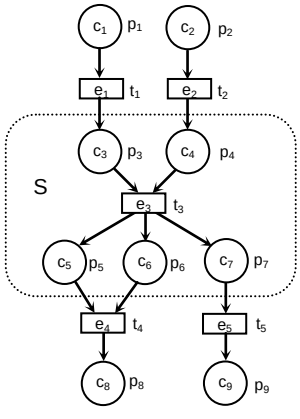


Figure 10. A uniquely labelled subnet S of a labelled net.

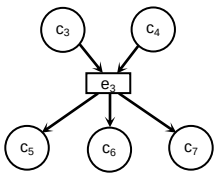


Figure 11. Pattern of S of the labelled net in Figure 10.

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 [39], 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 net is ultimately obtained.

The following algorithm formally describes the elimination process. A detailed elaboration of the elimination process can be found in the authors' earlier work [40].

Elimination of Duplicate Labels from a Labelled Net

1. Identify maximal disjoint groups of common subnets :
   - 1.1 Find all possible common subnets from N. Let $\Im = \{ L_1, L_2, ..., L_n \}$ be their patterns.
   - 1.2 Retain only the maximal patterns : Remove any L_i from $\Im$ if there exists $L_j \in \Im$ such that L_i is a sub-pattern of L_j and $\forall S_i \in [N, L_i], \exists S_j \in [N, L_j] : S_i$ is a subnet of S_j.
   - 1.3 Make the overlapping patterns disjoint : For every $L_i, L_j \in \Im$ such that $L_i \neq L_j$ and L_i and L_j are not disjoint, set $\Im = (\Im - \{ L_i, L_j \}) \cup \{ L_i \cap L_j \} \cup \{ L_i \setminus L_j \} \cup \{ L_j \setminus L_i \}$.
   - 1.4 Categorise the common subnets of N into groups $\{ [N, L_i], L_i \in \Im \}$.
2. For each group of common subnets $[N, L_i]$:
   - 2.1 Convert each subnet $S \in [N, L_i]$ 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 $p \in {}^\bullet t_i$: 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 $p \in t_j^\bullet$: 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 κ for each subnet $S \in [N, L_i]$:
     - 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 κ 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 κ to (p_j, t_j).
   - 2.3 Fuse the common subnets in $[N, L_i]$ into one single subnet.

We apply the algorithm for eliminating the duplicate labels for the integrated net (N, M_0) in Figure 9. Figure 12 shows the uniquely labelled net (N', M_0') so obtained.

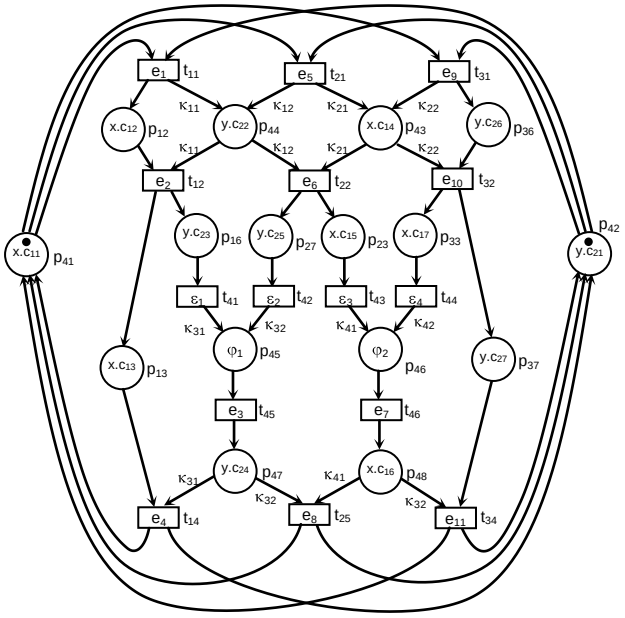


Figure 12. The uniquely labelled net obtained after eliminating duplicate labels from the integrated net in Figure 9.

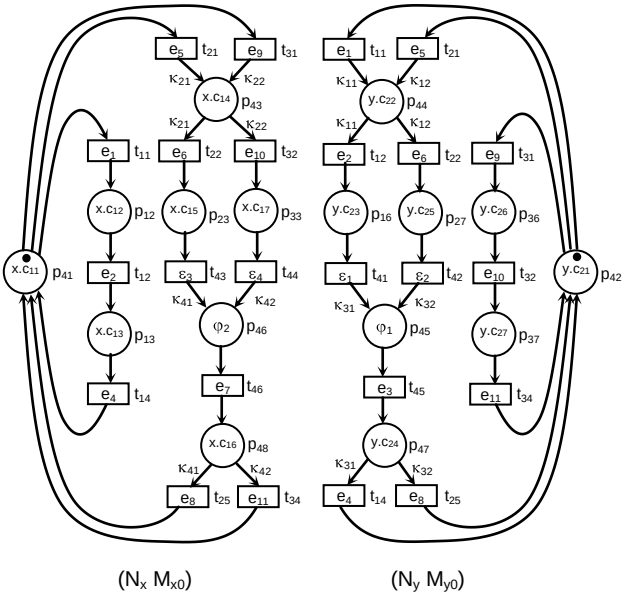


Figure 13. The nets obtained by projecting the integrated net in Figure 12 onto objects x and y.

7 Obtaining object-Based behavioural specifications

In this section, we show 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 (N', M_0') in Figure 12. 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.

Figure 13 shows the projections (N_x, M_{x0}) and (N_y, M_{y0}) obtained by projecting the net (N', M_0') in Figure 12 onto objects x and y, respectively. (N_x, M_{x0}) and (N_y, M_{y0}) are uniquely labelled, simply because (N', M_0') is uniquely labelled. They serve as the behavioural specifications for objects x and y, where conditions and events are uniquely represented.

8 Real-life example

This section presents a real-life example to further illustrate the refinement process.

The real-life example is an Office Access Control System. The system is briefly described as follows. It is a system used in a company for controlling staff accesses to its 30+ offices and laboratories. Among these offices and laboratories, some can be accessed by all staff while some others by authorised staff only and/or during specified time periods only. 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 centralised server. The server maintains the access privileges and validates every access to the offices/laboratories. There are three typical cases for each request for access.

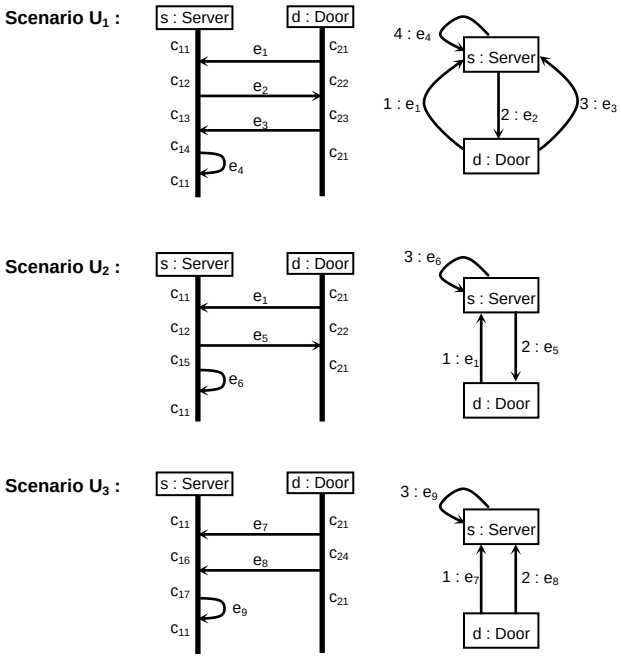
Authorised access (U_1). A staff member wants to access an office/laboratory. He/She presents his/her staff card via a card-reader. Access is granted. The door is unlocked for five seconds and then re-locked.

Unauthorised access (U_2). A staff member wants to access an office/laboratory. He/She presents his/her staff card via a card-reader. Access is not granted. The door is 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. There are three object interaction scenarios, corresponding to U_1, U_2 and U_3.

Figure 14 shows these object interaction scenarios specified as UML sequence diagrams and collaboration diagrams, where appropriate condition labels are appended for denoting the pre-conditions and post-conditions for each event occurrence.



Legends for condition labels :

c_{11}	Server is ready
c_{12}	Server is processing access request
c_{13}	Server is waiting for re-lock
c_{14}	Server is writing log (successful access)
c_{15}	Server is writing log (unsuccessful access)
c_{16}	Server is waiting for emergency reset
c_{17}	Server is writing log (emergency access)
c_{21}	Door is locked
c_{22}	Door is waiting for response
c_{23}	Door is unlocked (successful access)
c_{24}	Door is unlocked (emergency access)

Legends for event labels :

e_1	Request for access is received
e_2	Access is granted
e_3	Time expires after access granted
e_4	Successful access is committed
e_5	Access is not granted
e_6	Unsuccessful access is committed
e_7	Request for emergency access is received
e_8	Door is reset to normal
e_9	Emergency access is committed

Figure 14. Object interaction scenarios specified as UML sequence diagrams and collaboration diagrams (the Office Access Control System).

Step 1 of the proposed method is to specify object interaction scenarios as labelled nets. Figure 15 shows the labelled nets (N_1, M_{10}), (N_2, M_{20}) and (N_3, M_{30}) representing the object interaction scenarios for U_1, U_2 and U_3, respectively.

Step 2 of the proposed method is to synthesise the labelled net into an integrated system, and analyse the system on its liveness, boundedness, reversibility and conservativeness. (N_1, M_{10}), (N_2, M_{20}) and (N_3, M_{30}) are synthesised into an integrated net (N, 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}. Figure 16 shows the integrated net (N, M_0) so obtained.

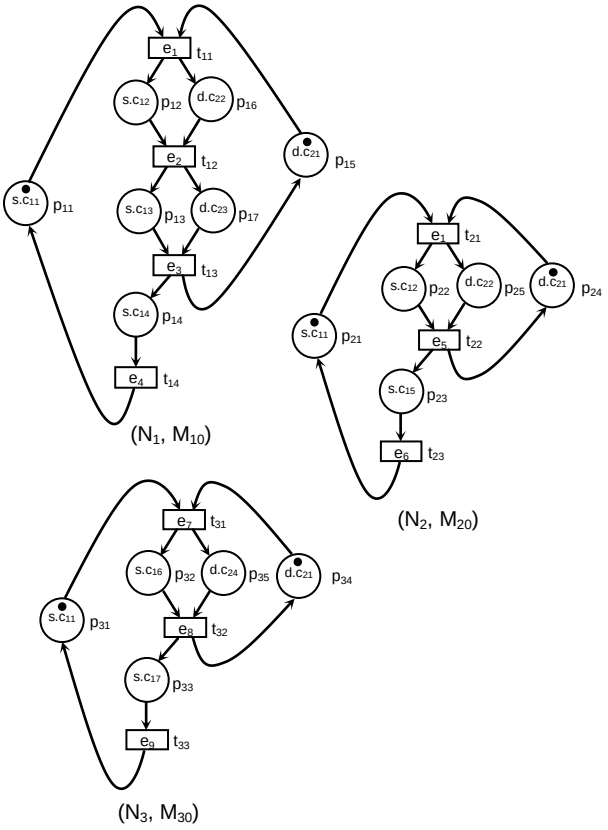


Figure 15. Labelled nets representing the object interaction scenarios in Figure 14.

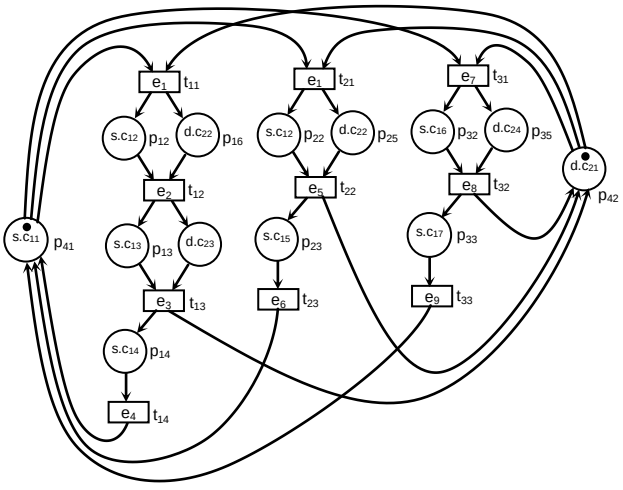


Figure 16. The integrated net obtained by synthesising the labelled nets in Figure 15.

The integrated net (N, M_0) is of an AMG-structure. Let R = { p_{41}, p_{42} }. For (N, M_0; R), every R-siphon contains a marked place and hence would never become empty. According to Properties 3.2 and 3.3, (N, M_0; R) is live and reversible. Since every place in its R-transform is covered by cycles, according to Property 3.4, (N, 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 Figure 16, (N, 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, in order for the integrated net to be effectively used for implementation purposes, these duplicate labels must be eliminated.

Step 3 of the proposed method is to eliminate duplicate condition labels and duplicate event labels from the integrated net (N, M_0) through the fusion of common subnets. We perform this elimination process by applying the algorithm described in Section 6. Figure 17 shows the uniquely labelled net (N', M_0') so obtained.

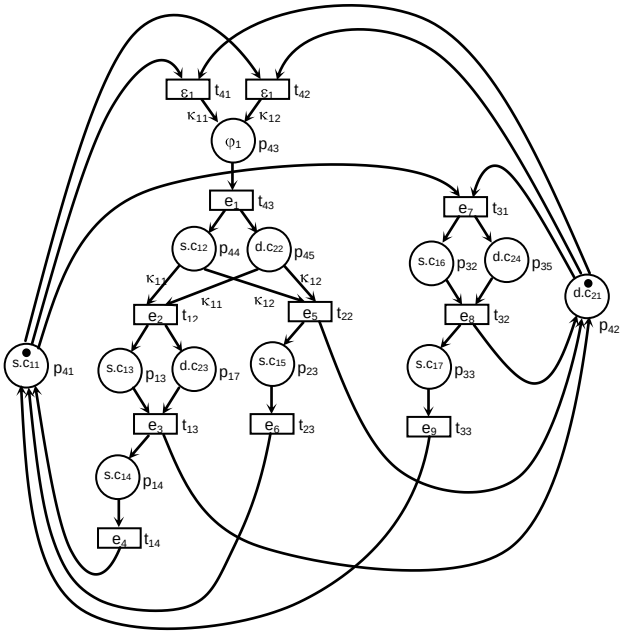


Figure 17. The uniquely labelled net obtained after eliminating duplicate labels from the integrated net in Figure 16.

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 (N', M_0') in Figure 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. Figure 18 shows the projections (N_s, M_{s0}) and (N_d, M_{d0}) obtained by projecting the integrated net (N', M_0') in Figure 17 onto objects s and d, respectively.

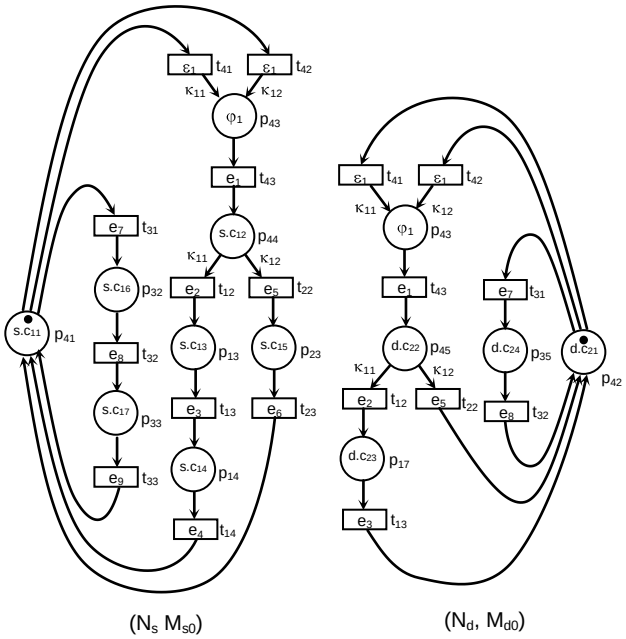


Figure 18. The nets obtained by projecting the integrated net in Figure 17 onto objects s and d.

As the integrated net (N', M_0') is uniquely labelled, its projections (N_s, M_{s0}) and (N_d, M_{d0}) are also uniquely labelled, where every condition or event is uniquely represented. (N_s, M_{s0}) and (N_d, M_{d0}) then serve as the behavioural specifications for the server (s : Server) and door (d : Door) objects, respectively.

9 Conclusion

One of the most difficult tasks in object-oriented system design is to obtain individual object-based behavioural specifications from a given set of object interaction scenarios. Not only conventional specification constructs for object interaction scenarios are too primitive to represent the partial ordering of events and the causal relationship between the events and conditions, there also involves different abstractions between intra-object lifecycle and inter-object interaction. Moreover, we have to ensure that the derived object-based behavioural specifications reflect exactly the given object interaction scenarios and that the system is well-behaved.

We proposed a Petri-net-based method for refining a given set of object interaction scenarios into individual object-based behavioural specifications. By specifying the object interaction scenarios as labelled nets with an AMG-structure and synthesising them into an integrated net, we analyse the system, based on the special properties of augmented marked graphs. For unique representation of events and conditions, an algorithm is applied to the integrated net to eliminate duplicate condition labels and event labels while preserving the event sequences. Object-based behavioural specifications are then obtained as projections of the integrated nets onto the objects. The whole refinement process has been described, elaborated and illustrated using the Office Access Control System example.

The proposed method offers a number of distinctive features. First, object interaction scenarios are formally specified as labelled nets which are unambiguous and semantically rich for an explicit representation of events, conditions and their causal relationships. Second, object-based behavioural specifications are rigorously derived from the given object interaction scenarios through systematic synthesis and projection. The behavioural specifications so obtained would reflect exactly the given object interaction scenarios. Third, liveness, boundedness, reversibility and conservativeness of the system can be effectively analysed by making use of the special properties of augmented marked graphs. Fourth, every event or condition is uniquely represented in the behavioural specifications so that the specifications can be readily used for implementation purposes.

With a strong theoretical foundation of Petri nets, the proposed method can be effectively used for refining object-based behavioural specifications from a set object interaction scenarios. It resolves a number of problems perplexing the designers of object-oriented systems, such as the lack of formality in specifying object interaction scenarios and the difficulty of ensuring the correctness of object behavioural specifications. The latter is especially important for systems involving shared resources, where erroneous situations such as deadlock and capacity overflow are easily induced. 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 perform rigorous system synthesis and analysis. The correctness of this refinement can be assured. Moreover, the object-based behavioural specifications so obtained can be readily used for code generation. This inevitably contributes towards a smooth transitions from functional requirements through design to implementation for object-oriented system development.

References

[1] J. Iivari (1995), "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.

[2] K.S. Cheung and K.O. Chow (1996), "Comparison of Object-Oriented Models by Views and Abstraction Constructs", *Proceedings of the International Conference on Intelligent Technologies in Human-Related Sciences*, pp. 335-342, Leon, Spain.

[3] K.O. Chow and S. Yeung (1996), "Behavioural Modelling in Object-Oriented Methodologies", *Information and Software Technology*, Vol. 38, No. 10, pp. 657-666.

[4] K.S. Cheung, K.O. Chow and T.Y. Cheung (1997), "A Feature-Based Approach for Consistent Object-Oriented Requirements Specifications". W.G. Wojtkowshi et al. (eds.), *Systems Development Methods for the Next Century*, pp. 31-38, Plenum Publishing.

[5] K.S. Cheung, K.O. Chow and T.Y. Cheung (1999), "Extending Formal Specification to Object Oriented Models Through Level-View Structured Schemas". J. Chen, J. Lu and B. Meyer (eds.), *Technology of Object Oriented Languages and Systems*, Vol. 31, pp. 118-125, IEEE Computer Society Press.

[6] B. Breu et al. (1998), "Systems, Views and Models of UML". M. Schader and A. Korthaus (eds.), *The Unified Modeling Language : Technical Aspects and Applications*, Physica-Verlag.

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

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

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

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

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

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

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

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

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

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

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

[18] K.S. Cheung, K.O. Chow and T.Y. Cheung (1998), "Consistency Analysis on Lifecycle Model and Interaction Model". C. Rolland and G. Grosz (eds.), *Object-Oriented Information Systems*, pp. 427-441, Springer.

[19] K.S. Cheung, K.O. Chow and T.Y. Cheung (1998), "Deriving Scenarios of Object Interaction through Petri Nets". J. Chen et al. (eds.), *Technology of Object Oriented Languages and Systems*, Vol. 27, pp. 118-125, IEEE Computer Society Press.

[20] M. Glinz (2000), "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.

[21] F. Bordeleau and R.J.A. Buhr (1997), "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.

[22] F. Bordeleau, J.P. Corriveau and B. Selic (2000), "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.

[23] B. Dano, H. Briand and F. Barbier (1996), "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.

[24] B. Dano, H. Briand and F. Barbier (1997), "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.

[25] P. Graubmann (1988), "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.

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

[27] K. Hiraishi (1992), "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.

[28] W.J. Lee, S.D. Cha and Y.R. Kwon (1998), "Integration and Analysis of Use Cases Using Modular Petri Nets in Requirement Engineering", *IEEE Transactions on Software Engineering*, Vol. 24, No. 12, pp. 1115-1130.

[29] J.L. Peterson (1981), *Petri Net Theory and the Modelling of System*, Prentice Hall.

[30] W. Reisig (1985), *Petri Nets : An Introduction*, Springer-Verlag.

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

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

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

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

[35] K.S. Cheung (2004), "New Characterisations for Live and Reversible Augmented Marked Graphs", *Information Processing Letters*, Vol. 92, No. 5, pp. 239-243.

[36] K.S. Cheung and K.O. Chow (2005), "Cycle Inclusion Property of Augmented Marked Graphs", *Information Processing Letters*, Vol. 94, No. 6, pp. 271-276.

[37] K.S. Cheung (2007), "Liveness and Boundedness of Augmented Marked Graphs", *IMA Journal of Mathematical Control and Information*, Vol. 24, No. 2, pp. 235-244.

[38] K.S. Cheung, T.Y. Cheung and K.O. Chow (2006), "A Petri-Net-Based Synthesis Methodology for Use-Case-Driven System Design", *Journal of Systems and Software*, Vol. 79, No. 6, pp. 772-790.

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

[40] K.S. Cheung and K.O. Chow (2006), "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.