Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 187 (2007) 161–172
www.elsevier.com/locate/entcs

# A Petri Net Based Method for Refining Object Oriented System Specifications

K.S. Cheung[1]

*Hong Kong Baptist University*
*Hong Kong*

K.O. Chow[2]

*City University of Hong Kong*
*Hong Kong*

**Abstract**

In object-oriented system design, requirements are given as use cases in the form of object interaction scenarios. One need to derive, from these object interaction scenarios, object-based behavioural specifications for implementation purposes. In this paper, a Petri-net-based method is proposed for the refinement process. The method starts with specifying the object interaction scenarios as labelled nets. These labelled nets are then synthesised into an integrated net. Duplicate labels are eliminated in order to attain a uniquely labelled net, on which object-based behavioural specifications are obtained as projections.

*Keywords:* Petri net, refinement, object-oriented system, object-oriented design, use case.

## 1 Introduction

In the past two decades, object orientation has been an influential discipline in software engineering. According to the principles of object orientation, a system is considered as a collection of objects which are interacting with others in order to accomplish the system functionalities. Conceptually, an object is an entity that encapsulates states and behaviours. It can be analysed in dual-aspects, namely, structure and behaviour. [1] [2] [3] [4] In the former, objects with the same attributes are grouped into classes while classes having common attributes are generalised to form an inheritance hierarchy. In the latter, objects exhibit different behaviours on interacting with others, thus demonstrating different object interaction scenarios. This paper primarily focus on the behavioural aspect of object-oriented system.

[1] Email: cheungks@hkbu.edu.hk
[2] Email: cspchow@cityu.edu.hk


doi:10.1016/j.entcs.2006.08.050

In object-oriented system design, the functional requirements of a system are given as a set of use cases - the typical cases or scenarios of how a system can be used. [5] [6] [7] [8] These use cases are expressed in terms of object interaction scenarios and formally specified as UML sequence diagrams and collaboration diagrams. [9] [10] [11] The system designer has to create, from these object interaction scenarios, individual-object-based specifications delineating the behaviours of individual objects. At least the following three problems have to be tackled in this refinement process.

*Specification constructs for object interaction scenarios being too primitive.* The UML sequence diagrams and collaboration diagrams lacks the formality for representing the pre-conditions and post-conditions of every event occurrence in an object interaction scenario. These are however required in deriving the object behavioural specification, where the conditions, events and their causal relationships need to be explicitly stated.

*Different abstraction between intra-object lifecycle and inter-object interaction.* It is very difficult to derive individual object behaviours (within the object lifecycle) from the object interaction scenarios (among multiple interacting 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 specification.* The object behavioural specification so derived should be correct in the sense that they reflect exactly the object interaction scenarios. [12] [13] [14] There should be no unintended interaction scenarios. In practice, without a formal method, the designer need to go through all possible object interaction scenarios. The process is time-consuming.

Based on our earlier related works, [15] [16] [17] in this paper, we propose a formal method for refining a given set of object interaction scenarios into the object-based behavioural specifications readily for implementation purposes, where the above mentioned problems can be resolved effectively. The proposed method is based on Petri nets.

The proposed method involves the following steps :

Step 1 : Each object interaction scenario is specified as a labelled Petri net (labelled net). These labelled nets are then synthesised into an integrated net.

Step 2 : Duplicate labels are eliminated from the integrated net, while preserving the firing sequences (event sequences).

Step 3 : Individual object-based specifications are obtained as projections of the integrated net on to the individual objects.

The rest of this paper is organised as follows. In Section 2, we show the specification of object interaction scenarios as labelled nets and the synthesis of these labelled nets (Step 1). Section 3 describes the elimination of duplicate labels of the integrated net so obtained (Step 2). In Section 4, we show how the individual object behavioural specification can be obtained through projection (Step 3). Section 5 is the conclusion. It should be noted that readers of this paper are expected to have

knowledge on Petri nets. [18] [19]

Throughout this paper, an Office Access Control System (OACS) is used for illustration. The OACS is used in a high-tech 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. For access control, every office/laboratory entrance is implemented with a card-reader, an emergency switch and an electronic lock, all connected by a centralised system server. The system server maintains the access privileges and validates every access to offices/laboratories. Basically, there are three possible cases (use cases) for each request for access.

$U_1$ : A staff member presents his/her staff card via a card-reader. Access is granted. The door is unlocked for five seconds and then re-locked.

$U_2$ : A staff member presents his/her staff card via a card-reader. Access is not granted. The door remains locked.

$U_3$ : A staff member presses the emergency key, and the door is unlocked immediately. After resetting by a security officer, the door is re-locked.

From the object-oriented perspectives, the server and doors are objects of the system. The use cases can be elaborated as object interaction scenarios between the server object and door object, and are specified as UML sequence diagrams and collaboration diagrams as shown in Figure 1. Condition labels are appended to these diagrams to denote the pre-conditions and post-conditions for each event occurrence.

## 2 Specifying Object Interaction Scenarios as Labelled Nets

In this section, after introducing labelled Petri nets (labelled nets), we show how object interaction scenarios can be specified as labelled nets. Then, we show the integration of these labelled nets.

**Definition 2.1** 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 2.2** 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 2 shows a labelled net. Places $p_3$, $p_4$, $p_5$, $p_6$, $p_9$ and $p_{10}$ are uniquely labelled. Places $p_1$, $p_2$, $p_7$ 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 $p_8$. On the other hand, transitions $t_3$, $t_4$ and $t_5$

are uniquely labelled. Transitions $t_1$, $t_2$, $t_6$ 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 $t_7$. The labelled net 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 corresponding 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 object interaction scenarios as labelled nets. These labelled nets are synthesised into an integrated net by fusing those places which refer to the same initial states or conditions. Using the OACS example, the object interaction scenarios $U_1$, $U_2$ and $U_3$ are specified as labelled nets ($N_1$, $M_{10}$), ($N_2$, $M_{20}$) and ($N_3$, $M_{30}$), as shown in Figure 3. The labelled nets are then synthesised into an integrated net, by fusing those places which refer to the same initial states or conditions : $p_{11}$, $p_{21}$ and $p_{31}$ are fused into one place $p_{41}$, and $p_{15}$, $p_{24}$ and $p_{34}$ into $p_{42}$. Figure 4 shows the integrated net (N, $M_0$).

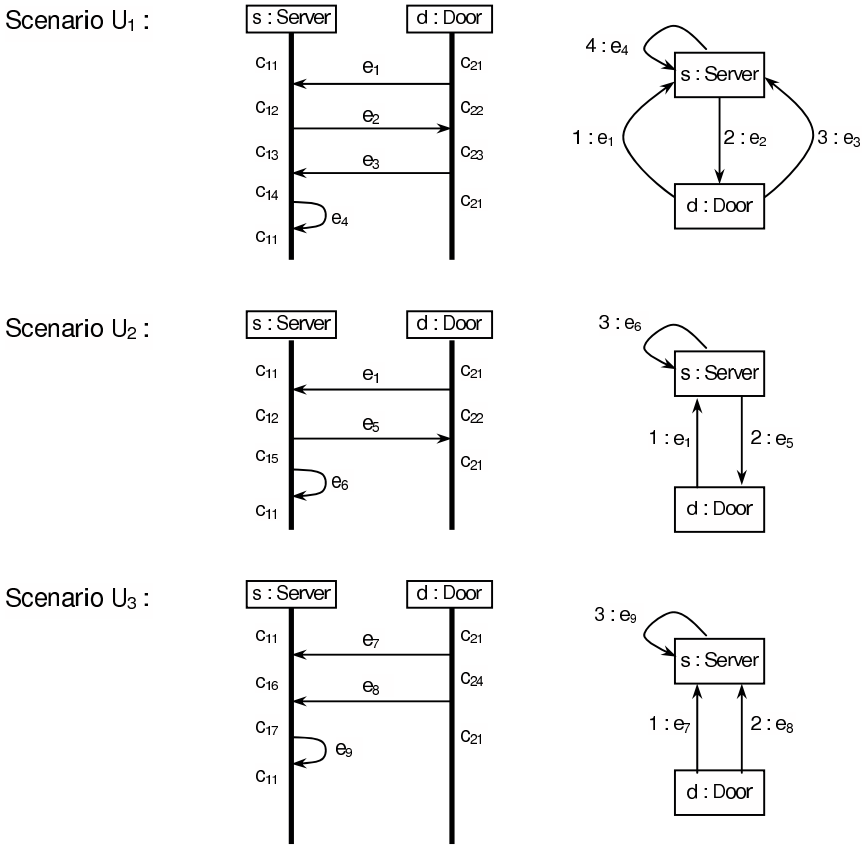
## 3 Eliminating Duplicate Labels of the Integrated Net

This section describes the elimination of duplicate labels of the integrated net. This refers to Step 2 of the proposed method.

Consolidating the object interaction scenarios, the integrated net obtained from Step 1 serves to represent the system as a whole. In general, it is not necessarily uniquely labelled. As in Figure 4, 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 $e_1$. This reflects the fact that the locations and conditions for executing the same event may be different at different moments. Yet, every condition is eventually implemented as a unique sub-state and every event as a unique operation. Therefore, in order for the integrated net to be effectively used for implementation purposes, duplicate condition labels and event labels must be eliminated.

A straight-forward strategy for this elimination is to fuse each set of places with the same condition label into a single place, and each set of transitions with the same event label into a single transition. However, this does not work because the resulting net may exhibit firing sequences different from the original ones. In other words, the system behaviours may be distorted. Hence, it is essentially required that the original firing sequences can be preserved after the fusion. Step 2 of the proposed method is to eliminate those duplicate labels while preserving the original firing sequences (event sequences).

**Definition 3.1** 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 identities of places and transitions of S.




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 |

Fig. 1. Object interaction scenarios specified as sequence diagrams (left) and collaboration diagrams (right).

For illustration, Figure 5 shows a uniquely labelled subnet of a labelled net while Figure 6 shows its pattern.

**Definition 3.2** 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 $L_x \cap L_y = \emptyset$.

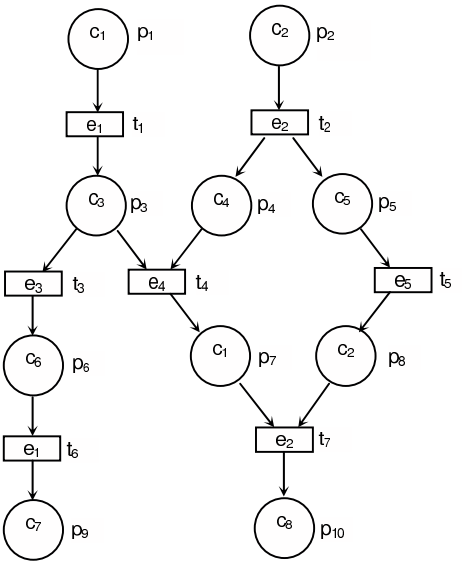



Fig. 2. A labelled Petri net (labelled net).

**Definition 3.3** 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 3.4** For a subnet S = $\langle$ P', T', F' $\rangle$ of a PT-net, Pre(S) = ($^\bullet$P'\T') $\cup$ ($^\bullet$T'\P') is called the pre-set of S, Post(S) = (P'$^\bullet$\T') $\cup$ (T'$^\bullet$\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 3.5** 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.

We propose to eliminate the duplicate labels by fusion of common subnets. The elimination process is outlined as follows.

*Identify group of common subnets for fusion.* These groups of common subnets need to be maximal and disjoint for two reasons. First, the net so 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.* In order to maintain firability of transitions and flow of tokens, the common subnets are transformed before fusion. Based on coloured Petri nets, [20], a unique colour is assigned to each common subnet (colour labels of its ingoing and outgoing arcs). A token flows into a common subnet is coloured according to the colour label of the ingoing arc. Its colour is reset as it flows out via the corresponding colour-labelled outgoing arc. Also, the subnets are converted to PP-type by appending dummy places and transitions.

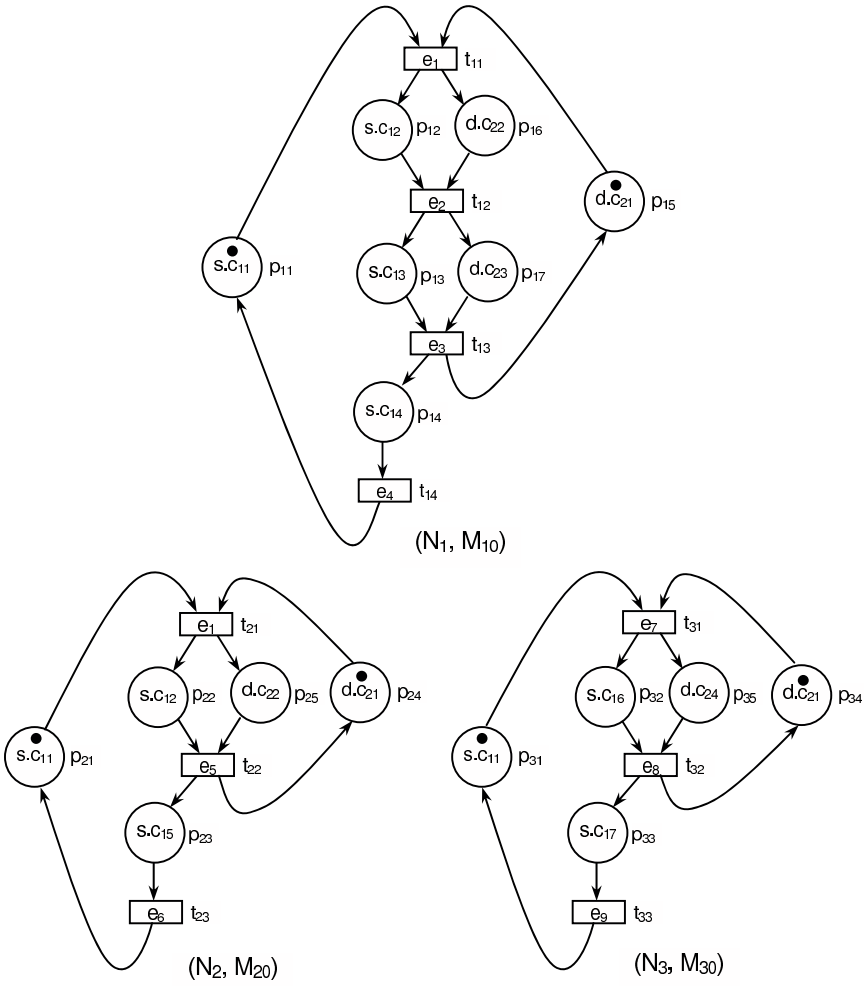



Fig. 3. Labelled nets representing the object interaction scenarios in Fig. 1.

*Fusion of transformed common subnets.* Then, the transformed common subnets of each group are fused into one single subnet. The labelled net so obtained after the fusion becomes uniquely labelled.

Algorithm 1 formally describes the elimination process as an algorithm. We apply the algorithm for eliminating the duplicate labels for the integrated net (N, $M_0$) in Figure 4. Figure 7 shows the uniquely labelled net (N', $M_0$') so obtained.

# 4 Obtaining Object-Based Specifications by Projection

This section shows Step 3 of our proposed method to obtain the individual object-based behavioural specification as projections of the integrated net. The projection is made by ignoring those places, transitions and arcs which are irrelevant to the object concerned. Using the OACS example, for object s (the server object), we keep those places with object label s (including dummy places) and those transitions

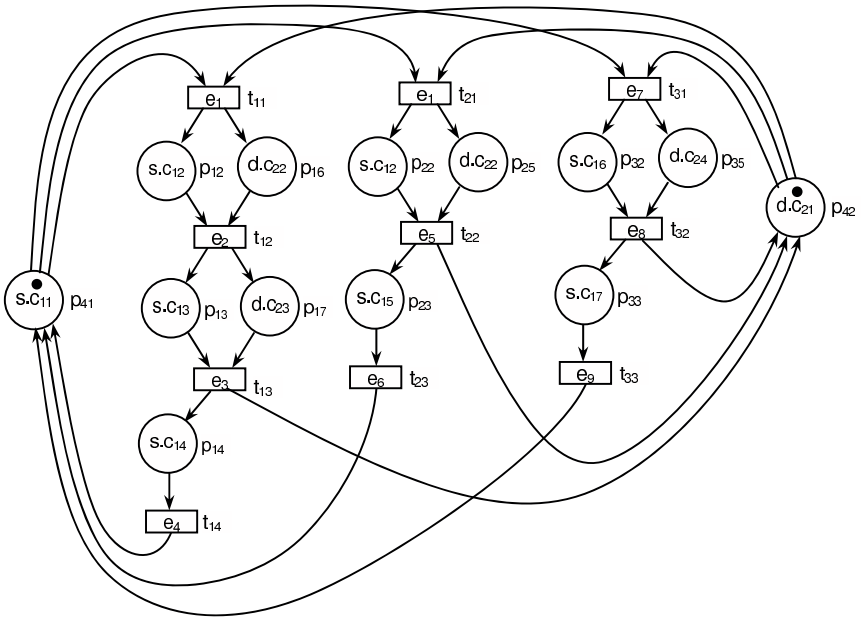



Fig. 4. The integrated net (N, $M_0$) obtained by synthesising the labelled nets in Fig. 3.

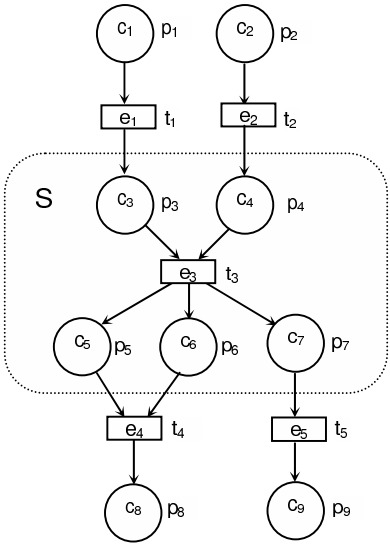



Fig. 5. A uniquely labelled subnet S of a labelled net.

having at least one input place or output place labelled by s. For object d (the door object), we keep those places with object label d (including dummy places) and those transitions having at least one input or output place labelled by d.

Figure 8 shows the projections ($N_s$, $M_{s0}$) and ($N_d$, $M_{d0}$) obtained by projecting the net (N', $M_0$') in Figure 7 onto objects s and d, respectively. Since (N', $M_0$') is uniquely labelled, both ($N_s$, $M_{s0}$) and ($N_d$, $M_{d0}$) are uniquely labelled.

($N_s$, $M_{s0}$) and ($N_d$, $M_{d0}$) serve to specify the behaviours of individual objects s and d, respectively. The specifications explicitly state the locations at which

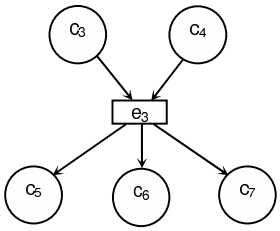



Fig. 6. The pattern of subnet S in Fig. 5.

**Algorithm 1** *Elimination of duplicate labels*
*Step 1 : Identify maximal disjoint groups of common subnets, as follows :*
*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* $L_i, L_j \in \Im$ *where* $L_i \neq L_j$ *and* $L_i$ *and* $L_j$ *are not disjoint, set* $\Im = (\Im - \{ L_i, L_j \}) \cup \{ L_i \cap L_j \} \cup \{ L_i \backslash L_j \} \cup \{ L_j \backslash L_i \}$*.*
*1.4 Categorise the common subnets of N into groups* $\{ [N, L_i], L_i \in \Im \}$*.*
*Step 2 : For each group of common subnets* $[N, L_i]$*, do the following :*
*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* $\varepsilon_i$*, dummy place* $p_i$*' with label* $\varphi_i$*, and arcs* $(t_i', p_i')$ *and* $(p_i', t_i)$*. (b) For each place* $p \in {}^\bullet t_i$ *: Remove arc* $(p, t_i)$*, and then create arc* $(p, t_i')$*. (c) Re-define S by including place* $p_i$*' and arc* $(p_i', t_i)$*.*
*2.1.2 For each transition* $t_j \in Tail(S)$ *: (a) Create dummy transition* $t_j$*' with unique label* $\varepsilon_j$*, dummy place* $p_j$*' with label* $\varphi_j$*, and arcs* $(t_j, p_j')$ *and* $(p_j', t_j')$*. (b) For each place* $p \in t_j^\bullet$ *: Remove arc* $(t_j, p)$*, and then create arc* $(t_j', p)$*. (c) Re-define S by including place* $p_j$*' and arc* $(t_j, p_j')$*.*
*2.2 Assign a unique colour label* $\kappa$ *for each subnet* $S \in [N, L_i]$ *:*
*2.2.1 For each arc* $(t_i, p_i)$ *such that* $t_i \in Pre(S)$ *and* $p_i \in Head(S)$ *: Assign colour label* $\kappa$ *to the arc* $(t_i, p_i)$*.*
*2.2.2 For each arc* $(p_j, t_j)$ *such that* $p_j \in Tail(S)$ *and* $t_j \in Post(S)$ *: Assign colour label* $\kappa$ *to the arc* $(p_j, t_j)$*.*
*2.3 Fuse the common subnets in* $[N, L_i]$ *into one single subnet.*

events occur and the locations at which conditions hold. The causal relationships between conditions and events are clearly delineated by flow relations between places and transitions. Many desirable properties, such as liveness, boundedness, safeness and reversibility, can be effectively analysed through many well-known analysis techniques on Petri nets. Besides, since the specifications are uniquely labelled net, conditions can be readily implemented as unique sub-states and events as unique operations.

# 5 Conclusion

In this paper, we proposed a Petri-net-based method for refining a given set of object interaction scenarios into individual object-based behavioural specifications and illustrate it using the OACS example. The proposed method has a number of distinctive features. First, the given object interaction scenarios are formally specified as labelled nets which are unambiguous and semantically rich. Second, the object-based behavioural specifications are rigorously derived from the object interaction scenarios through synthesis and projection. They reflect exactly the functionalities of the object interaction scenarios. Third, desirable properties, such as liveness and boundedness, can be analysed through many well-known analysis techniques on Petri nets. Fourth, the specifications contain no duplicate labels so that they can be readily used for implementation purposes.

With a strong theoretical foundation of Petri nets, the proposed method can be effectively used in object-oriented system design for deriving object-based behavioural specifications from a set of use cases given in the form of object interaction scenarios. It resolves a number of problems perplexing the designers of object-oriented systems, such as the lack of formality in the specifications of object interaction scenarios, the lack of rigorous and systematic approach to deriving

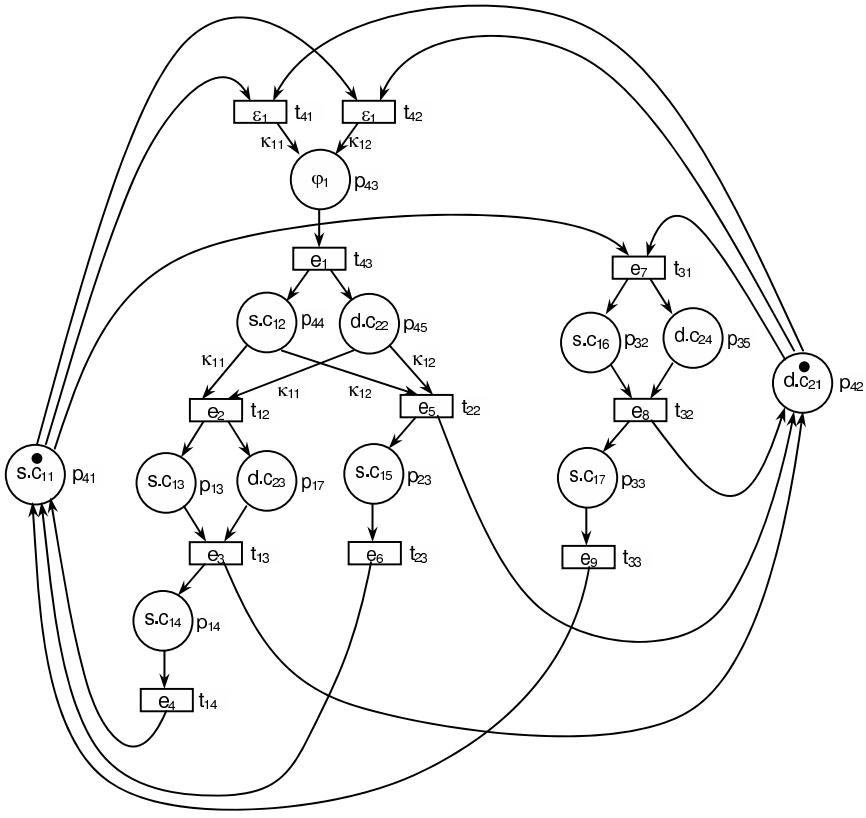



Fig. 7. The uniquely labelled net (N', $M_0$') so obtained after eliminating the duplicate labels from the integrated net (N, $M_0$) in Fig. 4.

object-based behavioural specifications from the object interaction scenarios, and the assurance of correctness of the derived specifications. As one further step, the proposed method can be implemented in object-oriented CASE tools to support object-oriented system design and use-case-driven system design.

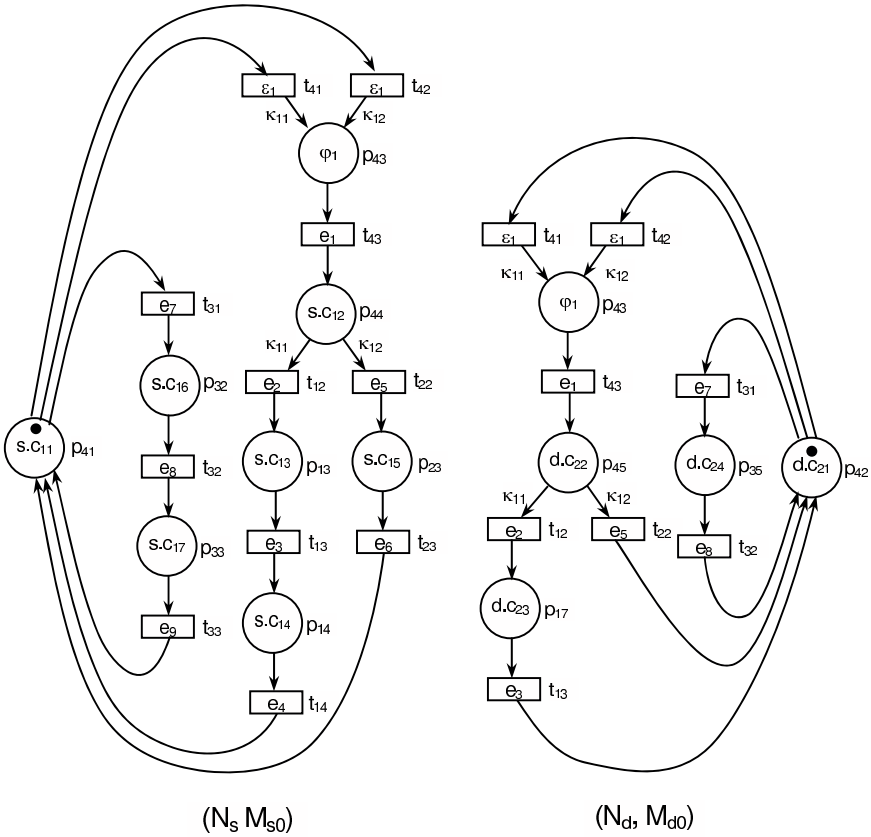



Fig. 8. The nets ($N_s$, $M_{s0}$) and ($N_d$, $M_{d0}$) obtained by projecting the integrated net (N', $M_0$') in Fig. 7 onto objects s and d.

# References

[1] Yourdon, E., *Object-Oriented Systems Design : An Integrated Approach*, Yourdon Press, 1994.

[2] Cheung, K.S. and Chow, K.O., "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, 1996.

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

[4] Graham, I., *Object-Oriented Methods : Principles and Practice*, Addison-Wesley, 2001.

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

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

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

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

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

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

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

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

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

[14] Glinz, M., "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.

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

[16] Cheung, K.S. and Chow, K.O., "A Synthesis Approach to Deriving Object-Based Specifications from Object Interaction Scenarios". In : Nilsson, A.G. et al. (Eds.), *Advances in Information Systems Development : Bridging the Gap between Academia and Industry*, pp. 647-656, Springer, 2006.

[17] Cheung, K.S. and Chow, K.O., "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.

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

[19] Murata, T., "Petri Nets : Properties, Analysis and Applications", *Proceedings of the IEEE*, Vol. 77, No. 4, 1989.

[20] Jensen, K., *Coloured Petri Nets : Basic Concepts, Analysis Methods and Practical Use*, Vol. 1, Springer-Verlag, 1992.