Computing and Informatics, Vol. 24, 2005, 629–653

A SYNTHESIS METHOD FOR DESIGNING SHARED-RESOURCE SYSTEMS

King Sing Cheung

SCE, Hong Kong Baptist University
Kowloon Tong, Hong Kong
e-mail: cheungks@hkbu.edu.hk

Manuscript received 18 November 2004; revised 14 November 2005
Communicated by José C. Cunha

Abstract. In system synthesis, one needs to derive from a given set of processes a system design which reflects exactly the functionalities of the processes and is free from erroneous situations such as deadlock and capacity overflow. This is especially important for shared-resource systems, in which errors are easily induced because of the sharing of common resources among different competing processes. In this paper, a synthesis method is proposed for designing shared-resource systems. It begins with specifying the given processes as augmented marked graphs. These augmented marked graphs are then synthesized through the fusion of common places which represents the shared resources. The net so obtained serves to represent the integrated system which reflects exactly the functionalities of the processes in the sense that the event sequences as well as the pre-conditions and post-conditions of each event occurrence are preserved. Based on the known properties of augmented marked graphs, the system properties such as liveness, boundedness and reversibility can be analysed effectively. The method is applied to manufacturing system design. Promising results are obtained.

Keywords: System synthesis, shared-resource system, Petri net, augmented marked graph

1 INTRODUCTION

Shared-resource systems generally refer to systems which possess some resources shared among different asynchronous processes. As the same resources are shared

among different competing processes, deadlocks may occur if the system is not carefully designed. Typically, a system is designed, basing on the functional requirements given by end-users as a set of processes. It is essentially required that, not only reflecting exactly the functionalities of the given processes, the design should also be correct in the sense that erroneous situations such as deadlock and capacity overflow would never occur.

The design process begins with elaborating the given processes as event traces or sequences. A system design is then derived from these event traces or sequences. In practice, without a rigorous and systematic method it is difficult to ensure that the design is correct and consistent, and verification is therefore required. For design correctness, it is necessary to check if any possible deadlock or capacity overflow would occur. For design consistency, all processes have to be walked through. The task is very time-consuming. In this paper, we propose a synthesis method to solve the problem effectively.

In the literature, there exist methods which derive a system design from a set of processes or event sequences. Graubmann proposed a method for constructing an elementary system from event traces, where the states and state transitions are deduced from the dependency among events [1]. Smith also proposed a method for constructing a condition-event system from a set of occurrence nets through the notion of quotient nets [2]. Hiraishi proposed a method for constructing a Petri net from a set of firing sequences, based on some dependency relation extracted from the firing sequences [3]. Bordeleau proposed a method that takes a traceable progression from use cases to object-based state machines [4, 5].

Chao introduced a synthesis method using knitting techniques [6, 7]. The starting point of Chao's method is a single process modelled by a set of close-loop sequentially-connected places and transitions with a marked home place. Processes are then appended in accordance with some synthesis rules, so that a live, bounded and reversible system can be obtained. Jeng proposed a synthesis method through the fusion of transitions and transition subnets [8, 9]. In Jeng's method, a system is obtained through the composition of modules represented by some specific nets called resource control nets. Interactions among resource control nets are represented by common transitions or transition subnets. The integrated net is bounded and conservative. A sufficient condition for structural liveness is derived.

Among the above methods, bottom-up synthesis is generally adopted in creating a system design from its processes, scenarios or component modules. Petri-net-based synthesis is the promising one that allows for rigorous analysis on the system properties. However, in many of these methods there is still a lack of formal and systematic procedures for deriving an integrated system design which is consistent with the processes or scenarios. Besides, the liveness, boundedness and reversibility of the outcoming system cannot be attained unless under some specific conditions and synthesis rules or procedures.

In this paper, based on augmented marked graphs, a method is proposed for synthesizing a shared-resource system from a given set of processes. The method

begins with specifying the processes as augmented marked graphs. These augmented marked graphs are synthesized through the fusion of common places which represents the shared resources. The net so obtained serves to represent the integrated system which reflects exactly the functionalities of the processes. The system properties can be analysed by making use of the known properties of augmented marked graphs. It will be shown how the method can be applied to the design of manufacturing systems which are typically shared-resource systems.

As a sub-class of Petri nets, augmented marked graphs possess a structure which is desirable for modelling shared resources. However, they have not been studied extensively. Chu first introduced augmented marked graphs and found some siphon-based properties pertaining to their liveness and reversibility [10]. We earlier proposed a number of new characterisations for live and reversible augmented marked graphs, where cycle-based characterisations were introduced [11, 12]. We now consolidate and extend the known properties of augmented marked graphs, develop a formal synthesis method where these properties can be effectively used for analysing the liveness, boundedness and reversibility of the outcoming system, and apply the synthesis method to manufacturing system design.

The rest of this paper is briefly structured as follows. Section 2 provides the preliminaries to be used in this paper. Section 3 introduces augmented marked graphs and their properties. In Section 4, the proposed synthesis method is described in details. Section 5 then presents its application to manufacturing system design. Section 6 concludes the results.

2 PRELIMINARIES

This section provides the preliminaries to be used in this paper for those readers who are not familiar with Petri nets [13–15].

A place-transition net (PT-net) is a bipartite graph consisting of two sorts of nodes called places and transitions, such that no arcs connect two nodes of the same sort. In graphical notation, a place is represented by a circle, a transition by a box, and an arc by a directed line. A Petri net is a PT-net where tokens are assigned to its places.

Definition 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 $W : F \to \{1, 2, \ldots\}$ 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 $\langle P, T, F \rangle$. In the rest of this paper, unless specified otherwise, all PT-nets are ordinary.

Definition 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 3. For a PT-net $N = \langle P, T, F, W \rangle$, a *path* is a sequence of nodes $\langle x_1, x_2, \ldots, x_n \rangle$, where $(x_i, x_{i+1}) \in F$ for $i = 1, 2, \ldots, n-1$. A path is said to be *elementary* if and only if it does not contain the same node more than once.

Definition 4. For a PT-net $N = \langle P, T, F, W \rangle$, a *cycle* is a sequence of places $\langle p_1, p_2, \ldots, p_n \rangle$ such that $\exists t_1, t_2, \ldots, t_n \in T : \langle p_1, t_1, p_2, t_2, \ldots, p_n, t_n \rangle$ forms an elementary path and $(t_n, p_1) \in F$.

For a PT-net where tokens are assigned to its places, the token distribution over its places is denoted by a marking.

Definition 5. For a PT-net $N = \langle P, T, F, W \rangle$, a *marking* is a function $M : P \to \{0, 1, 2, \ldots\}$, where $M(p)$ is the number of tokens in p. (N, M_0) represents N with an *initial marking* M_0.

Definition 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 7. For a PT-net (N, M_0), a sequence of transitions $\sigma = \langle t_1, t_2, \ldots, t_n \rangle$ is called a *firing sequence* if and only if $M_0\ [t_1\rangle \ldots [t_n\rangle M_n$. In notation, $M_0[N, \sigma\rangle M_n$ or $M_0[\sigma\rangle M_n$.

Definition 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).

Liveness, boundedness, safeness and reversibility are well known properties of Petri nets for describing the robustness of a system. Liveness implies deadlock freeness. Boundedness and safeness refer to the property that the system is free from capacity overflow. Reversibility refers to the capability of being reinitialised from any reachable state.

Definition 9. 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 10. For a PT-net (N, M_0), a place p is said to be *k-bounded* if and only if $\forall M \in [M_0\rangle : M(p) \leq k$, where k is a positive integer. (N, M_0) is said to be *bounded* if and only if every place is k-bounded, and *safe* if and only if every place is 1-bounded.

Definition 11. A PT-net (N, M_0) is said to be *reversible* if and only if $\forall M \in [M_0\rangle : M[*\rangle M_0$.

Figure 1 shows a PT-net (N, M_0), where every transition is live and every place is 1-bounded. (N, M_0) is live, bounded, safe and reversible.

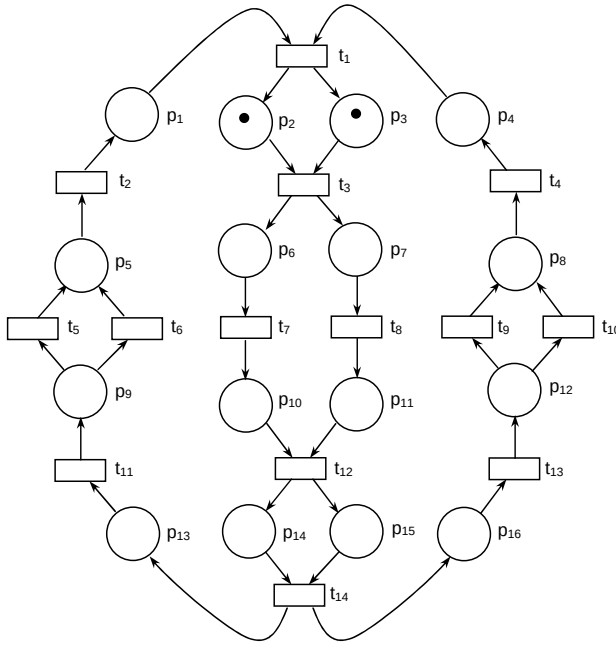

Fig. 1. A live, bounded, safe and reversible PT-net

3 AUGMENTED MARKED GRAPHS

This section describes augmented marked graphs and some major properties of augmented marked graphs reported in the literature.

Definition 12 ([10])**.** An *augmented marked graph* $(N, M_0; R)$ is a PT-net (N, M_0) with a specific subset of places R, satisfying the following conditions: (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 $r \in R$, there exist $k_r \geq 1$ pairs of transitions $D_r = \{\langle t_{s1}, t_{h1}\rangle, \langle t_{s2}, t_{h2}\rangle, \ldots, \langle t_{skr}, t_{hkr}\rangle\}$ such that $r^\bullet = \{t_{s1}, t_{s2}, \ldots, t_{skr}\} \subseteq T$ and ${}^\bullet r = \{t_{h1}, t_{h2}, \ldots, t_{hkr}\} \subseteq T$ 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 2 shows an augmented marked graph $(N, M_0; R)$, where $R = \{r_1, r_2\}$, $D_{r1} = \{\langle t_1, t_{10}\rangle, \langle t_2, t_8\rangle\}$ and $D_{r2} = \{\langle t_1, t_{10}\rangle, \langle t_3, t_9\rangle\}$.

Chu found a number of properties for augmented marked graphs, pertaining to their liveness and reversibility, based on siphons [10]. We earlier extended Chu's results and proposed new characterisations for live and augmented marked graphs, including a cycle-inclusion property [11, 12]. Huang investigated the composition of augmented marked graphs [16]. All these properties are summarised as follows.

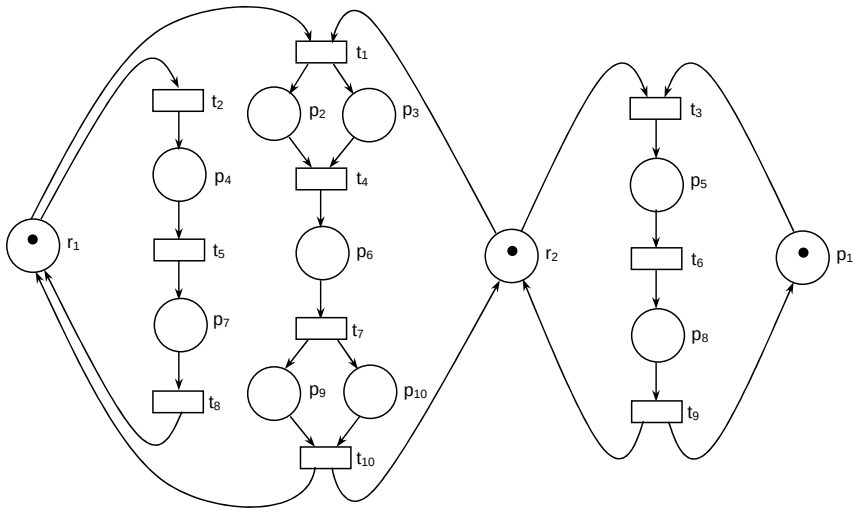

Fig. 2. An augmented marked graph

Definition 13. 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 another 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 tokens marked by M.

Definition 14. 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 another 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 a place marked by M.

Property 1 ([10])**.** An augmented marked graph is live if and only if it does not contain any potential deadlock. (Note: According to [10], a potential deadlock is a siphon which would eventually become empty.)

Property 2 ([10])**.** An augmented marked graph is reversible if it is live.

Property 3 ([10])**.** An augmented marked graph $(N, M_0; R)$ is live and reversible if every minimal siphon, which contains at least one place of R, contains a trap marked by M_0.

In our earlier works, we derived new characterisations for live and reversible augmented marked graphs [11, 12]. In particular, we introduced a cycle-inclusion property and proposed cycle-based characterisations which are different from the siphon-based characterisations. With the cycle-inclusion property, the checking of liveness and reversibility can be based on cycles instead of siphons.

Property 4 ([11]). An augmented marked graph $(N, M_0; R)$ is live and reversible if and only if no minimal siphons, which contain at least one place in R, eventually become empty.

For the augmented marked graph $(N, M_0; R)$, where $R = \{r_1, r_2\}$, shown in Figure 2, there are eight minimal siphons which contain r_1 or r_2 : $\{r_1, p_2, p_4, p_6, p_7, p_9\}$, $\{r_1, p_2, p_4, p_6, p_7, p_{10}\}$, $\{r_1, p_3, p_4, p_6, p_7, p_8\}$, $\{r_1, p_3, p_4, p_6, p_7, p_{10}\}$, $\{r_2, p_2, p_5, p_5, p_8, p_9\}$, $\{r_2, p_2, p_5, p_6, p_8, p_{10}\}$, $\{r_2, p_3, p_5, p_6, p_8, p_9\}$ and $\{r_2, p_3, p_5, p_6, p_8, p_{10}\}$. These minimal siphons would never become empty. According to Property 4, $(N, M_0; R)$ is live and reversible.

Definition 15. Let $N = \langle P, T, F, W\rangle$ be a PT-net. For a set of cycles $Y \subseteq \Omega_N$, $P[Y]$ denotes the set of places contained in Y. $T[Y] = {}^\bullet P[Y] \cap P[Y]^\bullet$ denotes the set of transitions generated by Y.

Definition 16 ([17]). For a PT-net $N = \langle P, T, F, W\rangle$, an elementary path $\rho = \langle x_1, x_2, \ldots, x_n\rangle$ is said to be *conflict-free* if and only if, for any transition x_i in ρ, $j \neq (i-1) \Rightarrow x_j \notin {}^\bullet x_i$.

Definition 17. For a PT-net $N = \langle P, T, F, W\rangle$, a set of cycle $Y \subseteq \Omega_N$ is said to be *conflict-free* if and only if, for any $q, q' \in P[Y]$, there exists in Y a conflict-free path from q to q'.

Figure 3 shows a PT-net $N = \langle P, T, F, W\rangle$. Consider $\gamma_1, \gamma_2, \gamma_3 \in \Omega_N[p_3]$, where $\gamma_1 = \langle p_3, p_2, p_7\rangle$, $\gamma_2 = \langle p_3, p_4\rangle$ and $\gamma_3 = \langle p_3, p_1, p_6, p_{10}, p_8\rangle$. $Y_1 = \{\gamma_1, \gamma_2\}$ is conflict-free as for any $q, q' \in P[Y_1]$, there exists in Y_1 a conflict-free path from q to q'. $Y_2 = \{\gamma_2, \gamma_3\}$ is not conflict-free. Consider $p_4, p_8 \in P[Y_2]$. p_4 is connected to p_8 via only one path $\rho = \langle p_4, t_5, p_3, t_1, p_1, t_3, p_6, t_6, p_{10}, t_9, p_8\rangle$ in Y_2, where ρ is not conflict-free because $p_4, p_8 \in {}^\bullet t_5$.

Definition 18 ([12]). For a PT-net $N = \langle P, T, F, W\rangle$, a place p is said to satisfy the *cycle-inclusion* property if and only if for any $Y \subseteq \Omega_N[p]$ such that Y is conflict-free, ${}^\bullet p \subseteq T[Y] \Rightarrow p^\bullet \subseteq T[Y]$.

For the PT-net $= \langle P, T, F, W\rangle$ shown in Figure 4, places p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11} and p_{12} satisfy the cycle-inclusion property. For example, for p_8, $\Omega_N[p_8] = \{\gamma_{81}, \gamma_{82}, \gamma_{83}, \gamma_{84}, \gamma_{85}\}$ where $\gamma_{81} = \langle p_8, p_1\rangle$, $\gamma_{82} = \langle p_8, p_2, p_4\rangle$, $\gamma_{83} = \langle p_8, p_2, p_9, p_1\rangle$, $\gamma_{84} = \langle p_8, p_1, p_5, p_9, p_2, p_4\rangle$ and $\gamma_{85} = \langle p_8, p_1, p_6, p_{10}, p_2, p_4\rangle$. For any $Y_8 \subseteq \Omega_N[p_8]$ such that Y_8 is conflict-free, ${}^\bullet p_8 = \{t_4\} \subseteq T[Y_8]$ and $p_8^\bullet = \{t_7\} \subseteq T[Y_8]$. Hence, p_8 satisfies the cycle-inclusion property. The same property applies to p_3, p_4, p_5, p_6, p_7, p_9, p_{10}, p_{11} and p_{12}.

Places p_1 and p_2 do not satisfy the cycle-inclusion property. For p_1, let $Y_1 = \{\gamma_{11}, \gamma_{12}\} \subseteq \Omega_N[p_1]$, where $\gamma_{11} = \langle p_1, p_8\rangle$ and $\gamma_{12} = \langle p_1, p_8, p_2, p_9\rangle$. Y_1 is conflict-free and $T[Y_1] = \{t_4, t_5, t_7, t_8\}$. Since ${}^\bullet p_1 = \{t_7, t_8\} \subseteq T[Y_1]$ and $p_1^\bullet = \{t_2, t_4\} \not\subseteq T[Y_1]$, p_1 does not satisfy the cycle-inclusion property. For p_2, let $Y_2 = \{\gamma_{21}, \gamma_{22}\} \subseteq \Omega_N[p_2]$, where $\gamma_{21} = \langle p_2, p_9\rangle$ and $\gamma_{22} = \langle p_2, p_9, p_1, p_8\rangle$. Y_2 is conflict-free and $T[Y_2] =$

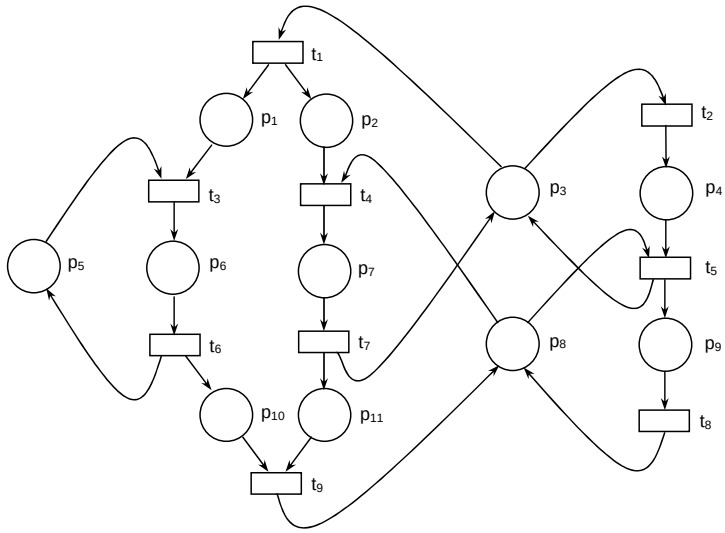

Fig. 3. Illustration of Conflict-Free Cycles

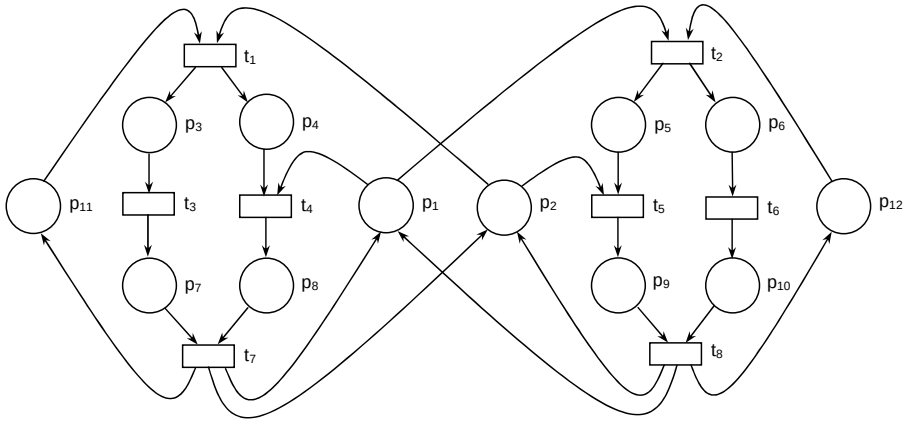

Fig. 4. Illustration of Cycle-Inclusion Property

$\{t_4, t_5, t_7, t_8\}$. Since ${}^\bullet p_2 = \{t_7, t_8\} \subseteq T[Y_2]$ and $p_2^\bullet = \{t_1, t_5\} \underline{\not\subset} T[Y_2], p_2$ does not satisfy the cycle-inclusion property.

Property 5 ([12])**.** An augmented marked graph $(N, M_0; R)$ is live and reversible if every place in R satisfies the cycle-inclusion property.

For the augmented marked graph $(N, M_0; R)$, where $R = \{r_1, r_2\}$, shown in Figure 2, both r_1 and r_2 satisfy the cycle-inclusion property. By Property 5, $(N, M_0; R)$ is live and reversible.

Property 6. Let $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ be two augmented marked graphs, where $R_1' = \{r_{11}, r_{12}, \ldots, r_{1k}\} \in R_1$ and $R_2' = \{r_{21}, r_{22}, \ldots, r_{2k}\} \in R_2$ are the common places that r_{11} and r_{21} are to be fused into one single place r_1, r_{12} and r_{22} into $r_2, \ldots, r_{1k}$ and r_{2k} into r_k. Then, the resulting net obtained after the fusion is also an augmented marked graph $(N, M_0; R)$, where $R = (R_1 \backslash R_1') \cup (R_2 \backslash R_2') \cup \{r_1, r_2, \ldots, r_k\}$ (obvious).

Property 7 ([16])**.** Let $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ be two augmented marked graphs, where $\{r_{11}, r_{12}, \ldots, r_{1k}\} \in R_1$ and $\{r_{21}, r_{22}, \ldots, r_{2k}\} \in R_2$ are the common places that r_{11} and r_{21} are to be fused into one single place r_1, r_{12} and r_{22} into $r_2, \ldots, r_{1k}$ and r_{2k} into r_k. The augmented marked graph $(N, M_0; R)$ obtained after the fusion is bounded if and only if $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ are bounded.

Property 8 ([16])**.** Let $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ be two augmented marked graphs, where $\{r_{11}, r_{12}, \ldots, r_{1k}\} \in R_1$ and $\{r_{21}, r_{22}, \ldots, r_{2k}\} \in R_2$ are the common places that r_{11} and r_{21} are to be fused into one single place r_1, r_{12} and r_{22} into $r_2, \ldots, r_{1k}$ and r_{2k} into r_k. Let $(N, M_0; R)$ be the augmented marked graph obtained after the fusion. $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ are live (respectively, reversible) if $(N, M_0; R)$ is live (respectively, reversible). Equivalently, $(N, M_0; R)$ is non-live (respectively, non-reversible) if $(N_1, M_{10}; R_1)$ or $(N_2, M_{20}; R_2)$ is non-live (respectively, non-reversible).

Figure 5 shows two augmented marked graphs $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$. $(N_2, M_{20}; R_2)$ is live and reversible while $(N_1, M_{10}; R_1)$ is not. Besides, both $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ are bounded. Suppose common places $r_{11} \in R_1$ and $r_{21} \in R_2$ are fused into one single place r. Figure 6 shows the augmented marked graph $(N, M_0; R)$ obtained after the fusion. $(N, M_0; R)$ is neither live nor reversible. Besides, it is bounded.

4 THE SYNTHESIS METHOD

This section describes a synthesis method for deriving from a given set of processes a system design for shared-resource systems. Dijkstra's dining philosopher problem will be used for illustration.

Typically, in the design of a shared-resource system the system requirements are given as a set of processes. A process is usually expressed as a sequence of events where each event occurrence is guarded by a set of pre-conditions and a set of post-conditions. It portrays an execution scenario for the system to accomplish a specific functionality. It starts at the system initial idle state with the availability of necessary resources, and returns to the idle state and releases the resources upon completion.

Based on a set of processes, a system is designed and implemented. It is essentially required that the system design should be consistent with respect to the given processes in the sense that the system reflects exactly the functionalities of these processes. In other words, there should not exist unrealistic processes – the ones

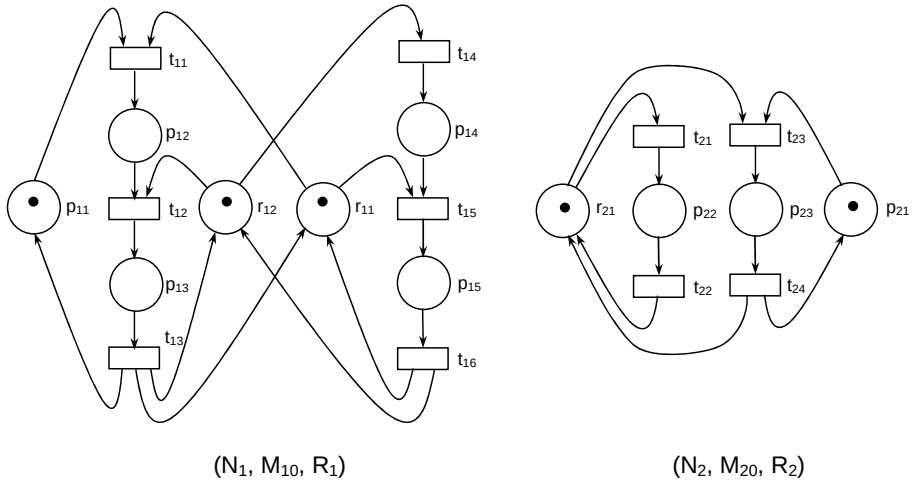

Fig. 5. Two augmented marked graphs to be fused

which are intended but not reflected in the system design. Also, there should not exist unintended processes – the ones which are not intended but reflected in the system design.

We propose a synthesis method to derive a system design from a set of processes which are assumed to be accurate and complete in the sense that these processes collectively describe all possible execution scenarios of the system. The method has two steps, as outlined in Figure 7. In brief, Step 1 is to specify the processes as augmented marked graphs. Step 2 is to synthesise these augmented marked graphs into one single integrated net through the fusion of common places which semantically denote the shared resources. The integrated net so obtained serves to represent the integrated system.

The synthesis method has the following distinctive features:

(i) Processes are formally specified as augmented marked graphs which possess some desirable structural characteristics for representing shared resources.

(ii) The integrated system obtained after the synthesis is also an augmented marked graph. Its liveness, boundedness and reversibility can be effectively analysed, basing on a number of known properties of augmented marked graphs.

(iii) The integrated system reflects exactly the functionalities of the given processes in the sense that the event sequences as well as the pre-conditions and post-conditions of each event occurrence are preserved.

Details of the synthesis method are described as follows. The synthesis method begins with specifying the given processes as augmented marked graphs. As mentioned earlier, the system requirements are given by end-users as processes. These processes portray the execution scenarios in which specific system functionalities are

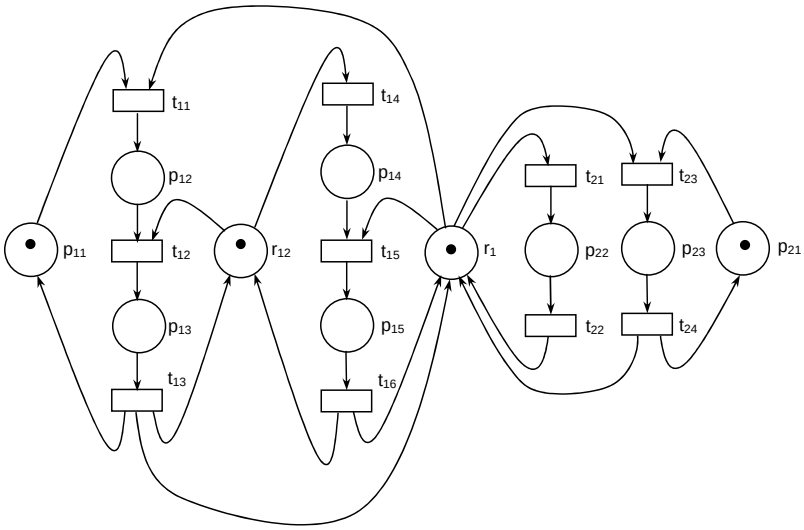

Fig. 6. An augmented marked graph obtained by fusing two augmented marked graphs in Figure 5

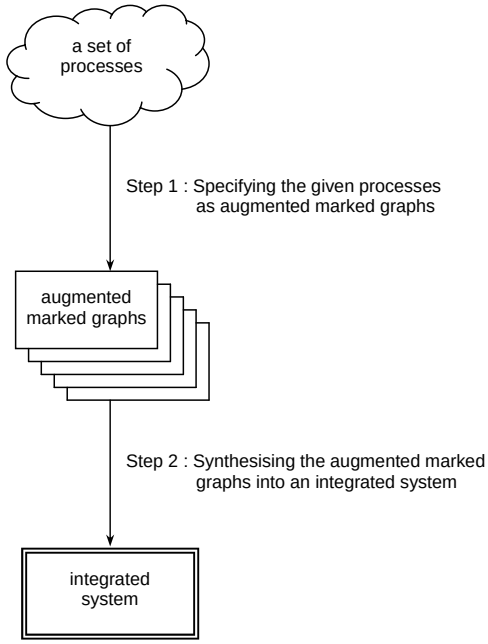

Fig. 7. Outline of the synthesis method

accomplished. Formally, a process is a collection of partially ordered event occurrences, each guarded by a set of pre-conditions and a set of post-conditions. Events and conditions are the essential elements, and their causal relationships characterise the processes.

For a process specified as an augmented marked graph $(N, M_0; R)$, the location where an event occurs is represented by a transition and the location of a condition by a place. For an event to occur, some conditions must be fulfilled in advance and some afterwards. They correspond to the pre-set and post-set of the transition representing that event. A process is specified as an augmented marked graph as follows. For each event occurrence in the process, a transition is created for the location of occurrence. Input and output places are created for its pre-conditions and post-conditions. Places for the shared resources are identified to form R. The initial marking M_0 represents the initial idle state. Execution begins at this initial marking and ends at the same marking.

After specifying the processes as augmented marked graphs, we synthesise these processes into an integrated system. In principle, a process portrays the partial system behaviours for a scenario of how a system is executed. These augmented marked graphs are basically partial system specifications which are to be synthesised altogether to form a complete system specification. Such synthesis process is aimed to derive a system specification by integrating these partial specifications into a single coherent whole. Given a set of processes specified as augmented marked graphs, we synthesise the augmented marked graphs through the fusion of common places which represent the shared resources.

In a shared-resource system, the same resource may be shared by different processes. Hence, among the augmented marked graphs which represent these processes, there exist common places for the same resources. From the system perspective, these common places logically refer to the same shared resources, and thus need to be fused. After fusing the corresponding common places, an integrated net is obtained. It serves to represent the integrated system. According to Property 6, the integrated net itself is also an augmented marked graph. Hence, based on the known properties of augmented marked graphs, the system can be analysed on its liveness, boundedness and reversibility.

In the following, we use the well-known Dijkstra's dining philosopher problem to illustrate the synthesis method. Example 1 shows the dining philosopher problem, where deadlocks never occur. Example 2 shows the dining philosopher problem with minor modifications, where deadlocks may occur. In each example, we show the specification of processes as augmented marked graphs and the synthesis of these processes into an integrated system, and then analyse the properties of the integrated system.

Example 1. The Dining Philosopher Problem
Six philosophers (H_1, H_2, H_3, H_4, H_5 and H_6) are sitting around a circular table for dinner. They are either meditating or eating the food at the centre of the table. There are six pieces of chopsticks (C_1, C_2, C_3, C_4, C_5 and C_6) shared among them

for getting the food to eat, as shown in Figure 8. For a philosopher to get the food to eat, both the chopstick at the right hand side and the chopstick at the left hand side must be available. The philosopher then grasps both chopsticks simultaneously and takes the food to eat. Afterwards, the chopsticks are released and returned to their original positions simultaneously. There are six processes U_1, U_2, U_3, U_4, U_5 and U_6 as follows.

U_1: H_1 grasps C_1 and C_2 once both C_1 and C_2 are available. H_1 gets the food to eat and then returns C_1 and C_2.

U_2: H_2 grasps C_2 and C_3 once both C_2 and C_3 are available. H_2 gets the food to eat and then returns C_2 and C_3.

U_3: H_3 grasps C_3 and C_4 once both C_3 and C_4 are available. H_3 gets the food to eat and then returns C_3 and C_4.

U_4: H_4 grasps C_4 and C_5 once both C_4 and C_5 are available. H_4 gets the food to eat and then returns C_4 and C_5.

U_5: H_5 grasps C_5 and C_6 once both C_5 and C_6 are available. H_5 gets the food to eat and then returns C_5 and C_6.

U_6: H_6 grasps C_6 and C_1 once both C_6 and C_1 are available. H_6 gets the food to eat and then returns C_6 and C_1.

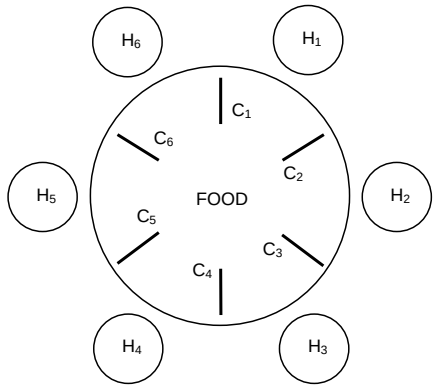

Fig. 8. The dinning philosopher problem

Figure 9 shows the augmented marked graphs $(N_1, M_{10}; R_1)$, $(N_2, M_{20}; R_2)$, $(N_3, M_{30}; R_3)$, $(N_4, M_{40}; R_4)$, $(N_5, M_{50}; R_6)$ and $(N_6, M_{60}; R_6)$ which represent U_1, U_2, U_3, U_4, U_5 and U_6, respectively. Table 1 lists the semantic meaning of the places and transitions. In particular, r_1, r_2, r_3, r_4, r_5 and r_6 represent the shared resources C_1, C_2, C_3, C_4, C_5 and C_6, respectively. For example, r_2 appears in (N_1, M_{10}, R_1) and (N_2, M_{20}, R_2), semantically meaning that C_2 is shared by processes U_1 and U_2. Hence, r_2 in (N_1, M_{10}, R_1) and r_2 in (N_2, M_{20}, R_2) are fused. Similar fusion apply to r_1, r_3, r_4, r_5 and r_6.

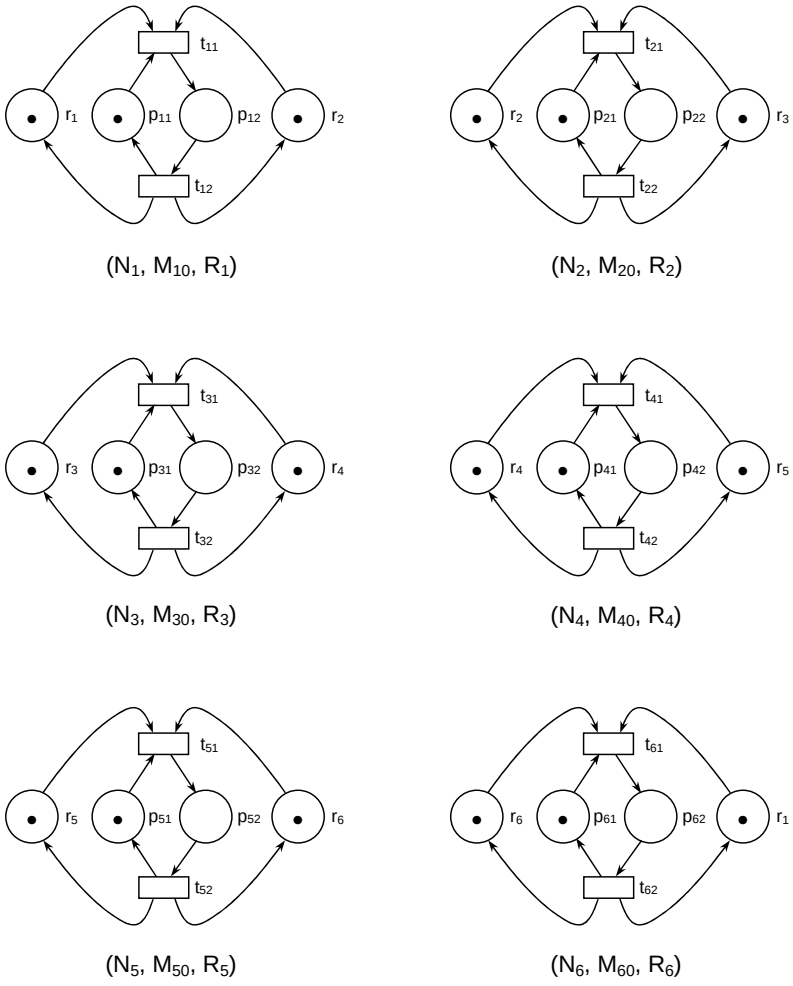

Fig. 9. Processes represented by augmented marked graphs (Example 1)

Figure 10 shows the integrated net $(N, M_0; R)$ obtained after fusing the corresponding common places. It serves to represent the integrated system which reflects exactly the functionalities of U_1, U_2, U_3, U_4, U_5 and U_6 in the sense that the event sequences (firing sequences) as well as the pre-conditions and post-conditions of each event occurrence are preserved. According to Property 6, $(N, M_0; R)$ is structurally an augmented marked graph. For $(N, M_0; R)$, every minimal siphon contains a marked trap, and thus would never become empty. According to Properties 3 or 4, it is live and reversible. Besides, since $(N_1, M_{10}; R_1)$, $(N_2, M_{20}; R_2)$, $(N_3, M_{30}; R_3)$, $(N_4, M_{40}; R_4)$, $(N_5, M_{50}; R_5)$ and $(N_6, M_{60}; R_6)$ are all bounded, according to Property 7, $(N, M_0; R)$ is bounded.

Semantic meaning for places		**Semantic meaning for transitions**	
p_{11}	H_1 is meditating.	t_{11}	H_1 takes the action to grasp C_1 and C_2.
p_{12}	H_1 has got C_1 and C_2 and takes the food.	t_{12}	H_1 takes the action to return C_1 and C_2.
p_{21}	H_2 is meditating.	t_{21}	H_1 takes the action to grasp C_2 and C_3.
p_{22}	H_2 has got C_2 and C_3 and takes the food.	t_{22}	H_1 takes the action to return C_2 and C_3.
p_{31}	H_3 is meditating.	t_{31}	H_1 takes the action to grasp C_3 and C_4.
p_{32}	H_3 has got C_3 and C_4 and takes the food.	t_{32}	H_1 takes the action to return C_3 and C_4.
p_{41}	H_4 is meditating.	t_{41}	H_1 takes the action to grasp C_4 and C_5.
p_{42}	H_4 has got C_4 and C_5 and takes the food.	t_{42}	H_1 takes the action to return C_4 and C_5.
p_{51}	H_5 is meditating.	t_{51}	H_1 takes the action to grasp C_5 and C_6.
p_{52}	H_5 has got C_5 and C_6 and takes the food.	t_{52}	H_1 takes the action to return C_5 and C_6.
p_{61}	H_6 is meditating.	t_{61}	H_1 takes the action to grasp C_6 and C_1.
p_{62}	H_6 has got C_6 and C_1 and takes the food.	t_{62}	H_1 takes the action to return C_6 and C_1.
r_1	C_1 is available for pick.		
r_2	C_2 is available for pick.		
r_3	C_3 is available for pick.		
r_4	C_4 is available for pick.		
r_5	C_5 is available for pick.		
r_6	C_6 is available for pick.		

Table 1. Semantic meaning for places and transitions (Example 1)

Example 2. The Modified Dining Philosopher Problem
The procedure for a philosopher to get the food to eat is now modified. A philosopher first grasps the chopstick at the right hand side if available, and then grasps the chopstick at the left hand side if available. Once both chopsticks are grasped, he or she can take the food to eat. Afterwards, the chopsticks are released and returned to their original positions simultaneously. Processes U_1, U_2, U_3, U_4, U_5 and U_6 are now modified as follows.

U_1: H_1 first grasps C_1 once C_1 is available. H_1 holds C_1 and grasps C_2 once C_2 is available. H_1 gets the food to eat and then returns C_1 and C_2.

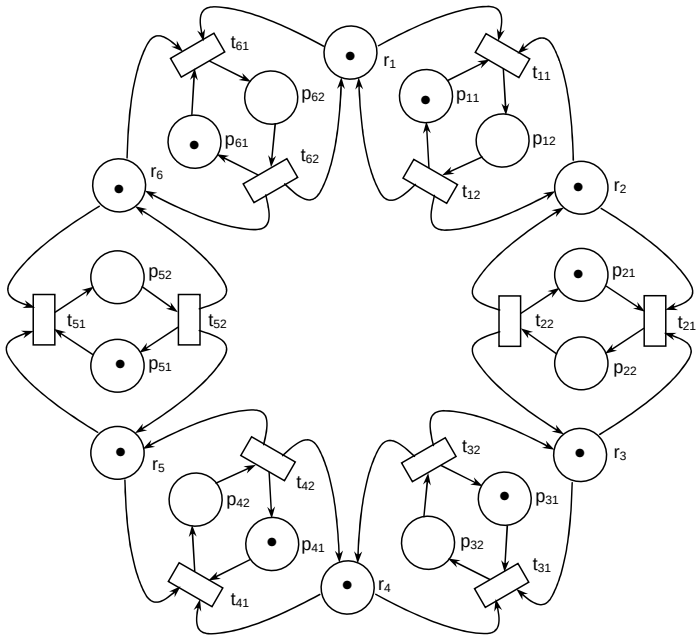

Fig. 10. The integrated system (Example 1)

U_2: H_2 first grasps C_2 once C_2 is available. H_2 holds C_2 and grasps C_3 once C_3 is available. H_2 gets the food to eat and then returns C_2 and C_3.

U_3: H_3 first grasps C_3 once C_3 is available. H_3 holds C_3 and grasps C_4 once C_4 is available. H_3 gets the food to eat and then returns C_3 and C_4.

U_4: H_4 first grasps C_4 once C_4 is available. H_4 holds C_4 and grasps C_5 once C_5 is available. H_4 gets the food to eat and then returns C_4 and C_5.

U_5: H_5 first grasps C_5 once C_5 is available. H_5 holds C_5 and grasps C_6 once C_6 is available. H_5 gets the food to eat and then returns C_5 and C_6.

U_6: H_6 first grasps C_6 once C_6 is available. H_6 holds C_6 and grasps C_1 once C_1 is available. H_6 gets the food to eat and then returns C_6 and C_1.

Figure 11 shows augmented marked graphs $(N_1, M_{10}; R_1)$, $(N_2, M_{20}; R_2)$, $(N_3, M_{30}; R_3)$, $(N_4, M_{40}; R_4)$, $(N_5, M_{50}; R_6)$ and $(N_6, M_{60}; R_6)$ which represent U_1, U_2, U_3, U_4, U_5 and U_6, respectively. Table 2 lists the semantic meaning of the places and transitions. Figure 12 shows the augmented marked graph $(N, M_0; R)$ obtained after fusing the corresponding common places. It serves to represent the integrated system. For $(N, M_0; R)$, the set of places $\{r_1, p_{13}, r_2, p_{23}, r_3, p_{33}, r_4, p_{43}, r_5, p_{53}, r_6, p_{63}\}$ is a siphon (minimal siphon) which would become empty after firing $\langle t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}\rangle$. According to Property 1, $(N, M_0; R)$ is not live. Deadlocks may occur, for example, after firing $\langle t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}\rangle$. Besides, as $(N_1, M_{10}; R_1)$, $(N_2, M_{20}; R_2)$,

$(N_3, M_{30}; R_3)$, $(N_4, M_{40}; R_4)$, $(N_5, M_{50}; R_6)$ and $(N_6, M_{60}; R_6)$ are all bounded, according to Property 7, $(N, M_0; R)$ is bounded.

5 APPLICATION TO MANUFACTURING SYSTEM DESIGN

A manufacturing system is typically a shared-resource system where the resources are so scarce that used to be maximally shared among different processes [18–24]. For this reason, different processes would compete for the same resources. This would, however, lead to deadlock if the system is not carefully designed. Thus, deadlock freeness and liveness are important properties for a manufacturing system. Besides, as resources are scarce and have finite and limited capacity, the system should be free from capacity overflow. Another equally important property for a manufacturing system is the capability of being reinitialised from any reachable state. These robust properties refer to the liveness, boundedness and reversibility of a system.

In this section, it will be shown how the synthesis method can be applied to the design of manufacturing systems, where the liveness, boundedness and reversibility of the outcome system can be analysed effectively.

Given a set of manufacturing processes, we derive a system design as follows. For each process, we construct an augmented marked graph by identifying the event occurrences and creating transitions for locations of these event occurrences and places for locations of the pre-conditions and post-conditions of each event occurrence. The shared-resources are represented by common places. These augmented marked graphs are then synthesised into one single integrated net through the fusion of the corresponding common places. The integrated net is also an augmented marked graph which serves to represent the integrated system that reflects exactly the functionalities of the processes. Its liveness, boundedness and reversibility can be analysed by making use of the known properties of augmented marked graphs. For illustration, we use the FWS-200 example [24, pp. 121–124].

Example 3. The FWS-200 Flexible Workstation System
The FWS-200 Flexible Workstation System is a manufacturing system for production of circuit boards. It consists of two robots R_1 and R_2, one feeder area and one PCB area, as shown in Figure 13 [24, pp. 121–124]. The robots repeatedly perform the activities of picking components from the feeder area, moving in the PCB area for inserting components, and finishing the product. The feeder area and PCB area are the shared resources. There are two manufacturing processes U_1 and U_2, as follows.

U_1: Robot R_1 picks components from the feeder area, moves into the PCB area for inserting. The finished product is then moved out from the PCB area.

U_2: Robot R_2 picks components from the feeder area, moves into the PCB area for inserting. The finished product is then moved out from the PCB area.

U_1 and U_2 are specified as augmented marked graphs $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$, respectively, as shown in Figure 14. Table 3 lists the semantic meaning of

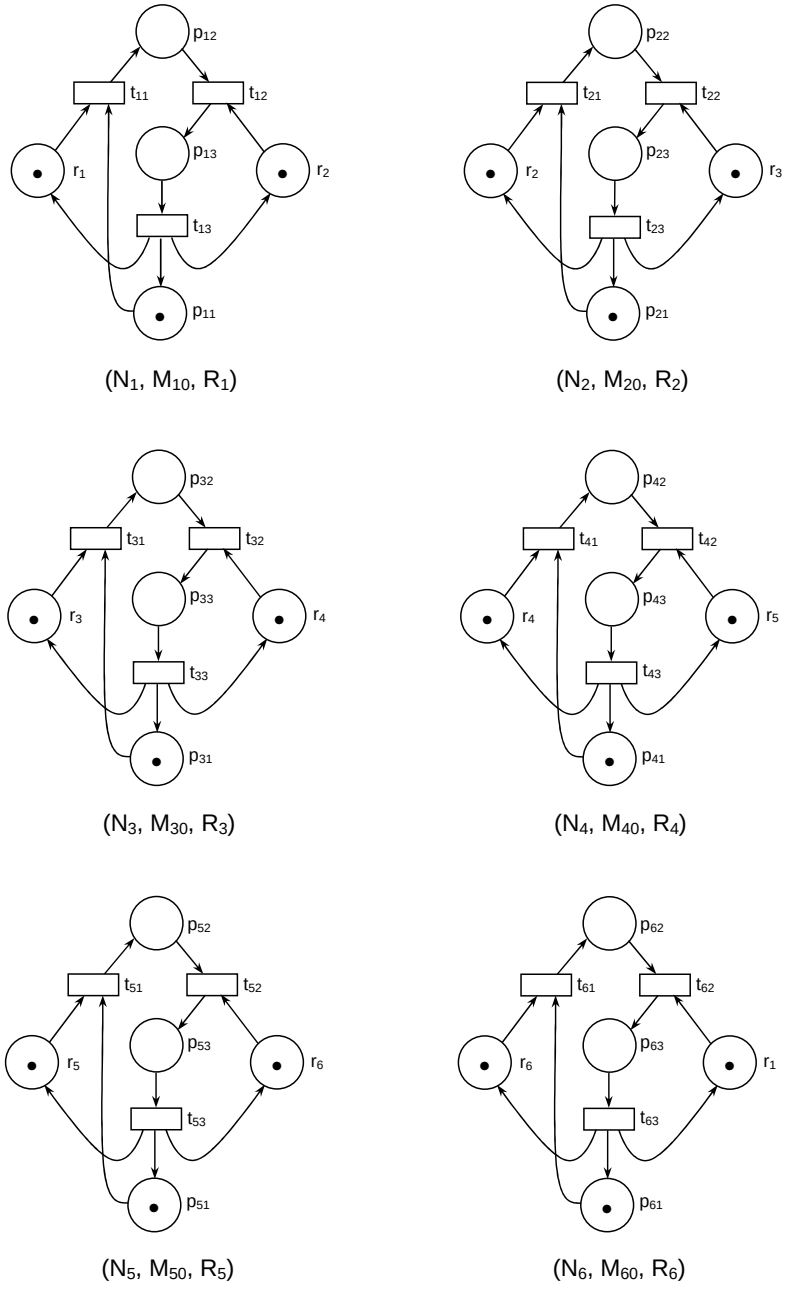

Fig. 11. Processes represented by augmented marked graphs (Example 2)

Semantic meaning for places		**Semantic meaning for transitions**	
p_{11}	H_1 is meditating.	t_{11}	H_1 takes the action to grasp C_1.
p_{12}	H_1 has got C_1 and prepares to pick C2.	t_{12}	H_1 takes the action to grasp C_2.
p_{13}	H_1 has got C_1 and C_2 and takes the food.	t_{13}	H_1 takes the action to return C_1 and C_2.
p_{21}	H_2 is meditating.	t_{21}	H_2 takes the action to grasp C_2.
p_{22}	H_2 has got C_2 and prepares to pick C_3.	t_{22}	H_2 takes the action to grasp C_3.
p_{23}	H_2 has got C_2 and C_3 and takes the food.	t_{23}	H_2 takes the action to return C_2 and C_3.
p_{31}	H_3 is meditating.	t_{31}	H_3 takes the action to grasp C_3.
p_{32}	H_3 has got C_3 and prepares to pick C_4.	t_{32}	H_3 takes the action to grasp C_4.
p_{33}	H_3 has got C_3 and C_4 and takes the food.	t_{33}	H_3 takes the action to return C_3 and C_4.
p_{41}	H_4 is meditating.	t_{41}	H_4 takes the action to grasp C_4.
p_{42}	H_4 has got C_4 and prepares to pick C_5.	t_{42}	H_4 takes the action to grasp C_5.
p_{43}	H_4 has got C_4 and C_5 and takes the food.	t_{43}	H_4 takes the action to return C_4 and C_5.
p_{51}	H_5 is meditating.	t_{51}	H_5 takes the action to grasp C_5.
p_{52}	H_5 has got C_5 and prepares to pick C_6.	t_{52}	H_5 takes the action to grasp C_6.
p_{53}	H_5 has got C_5 and C6 and takes the food.	t_{53}	H_5 takes the action to return C_5 and C_6.
p_{61}	H_6 is meditating.	t_{61}	H_6 takes the action to grasp C_6.
p_{62}	H_6 has got C_6 and prepares to pick C_1.	t_{62}	H_6 takes the action to grasp C_1.
p_{63}	H_6 has got C_6 and C_1 and takes the food.	t_{63}	H_6 takes the action to return C_6 and C_1.
r_1	C_1 is available for pick.		
r_2	C_2 is available for pick.		
r_3	C_3 is available for pick.		
r_4	C_4 is available for pick.		
r_5	C_5 is available for pick.		
r_6	C_6 is available for pick.		

Table 2. Semantic meaning for places and transitions (Example 2)

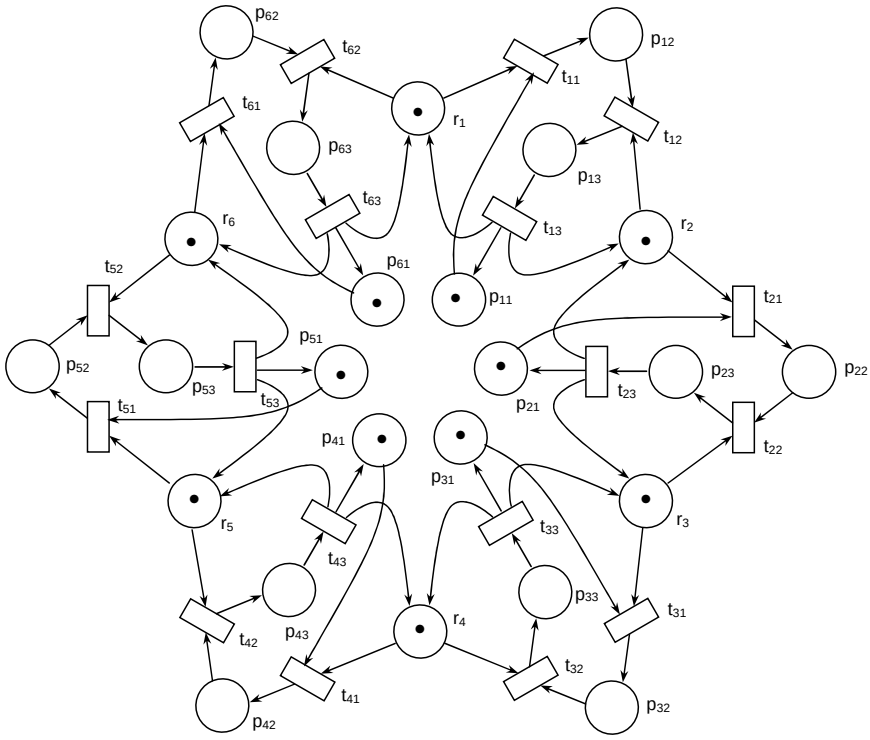

Fig. 12. The integrated system (Example 2)

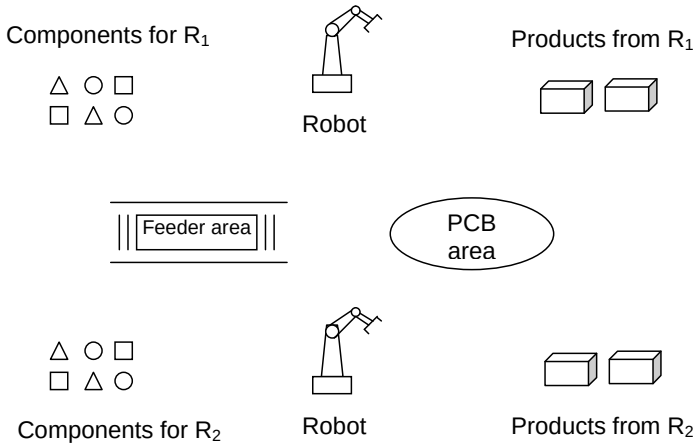

Fig. 13. The FWS-200 flexible workstation system

the places and transitions. Among $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$, there are two common places r_1 and r_2 which represent the feeder area and PCB area, respectively.

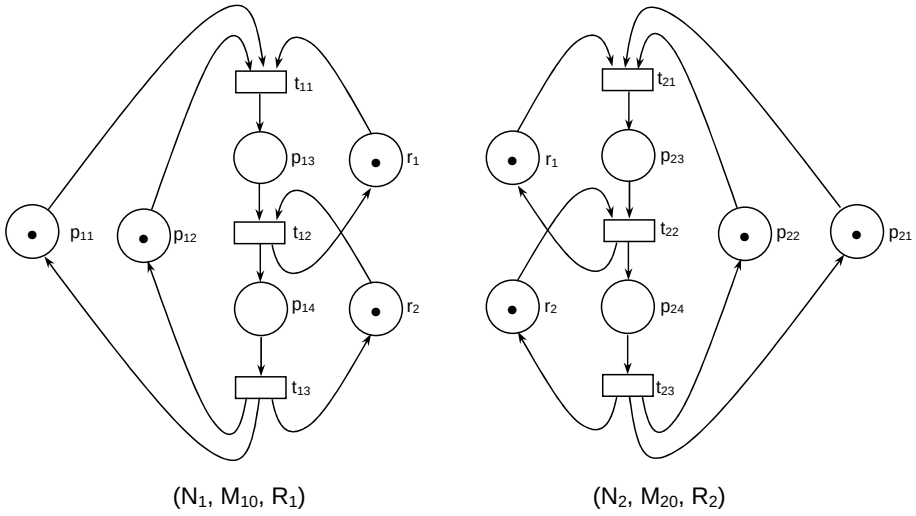

Fig. 14. Processes represented by augmented marked graphs (Example 3)

Semantic meaning for places		**Semantic meaning for transitions**	
p_{11}	R_1 is ready.	t_{11}	R_1 starts picking components.
p_{12}	Components for R_1 are available.	t_{12}	R_1 finishes picking components and starts inserting components.
p_{13}	R_1 is picking components from feeder.		
p_{14}	R_1 is inserting components in PCB area.	t_{13}	R_1 finishes inserting components and starts moving out the finished product.
p_{21}	R_2 is ready.		
p_{22}	Components for R_2 are available.	t_{21}	R_2 starts picking components.
p_{23}	R_2 is picking components from feeder.	t_{22}	R_2 finishes picking components and starts inserting components.
p_{24}	R_2 is inserting components in PCB area.		
r_1	Feeder area is available.	t_{23}	R_2 finishes inserting components and starts out the finished product.
r_2	PCB area is available.		

Table 3. Semantic meaning for places and transitions (Example 3)

Figure 15 shows the augmented marked graph $(N, M_0; R)$ obtained from $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ after fusing the corresponding common places. It serves

to represent the integrated system, which reflects exactly the functionalities of U_1 and U_2 in the sense that the event sequences and the pre-condition and post-condition of event occurrences are preserved. For $(N, M_0; R)$, every minimal siphon, which contains r_1 or r_2, contains a marked trap. According to Properties 3 and 4, $(N, M_0; R)$ is live and reversible. Besides, as $(N_1, M_{10}; R_1)$ and $(N_2, M_{20}; R_2)$ are bounded, according to Property 7, $(N, M_0; R)$ is bounded.

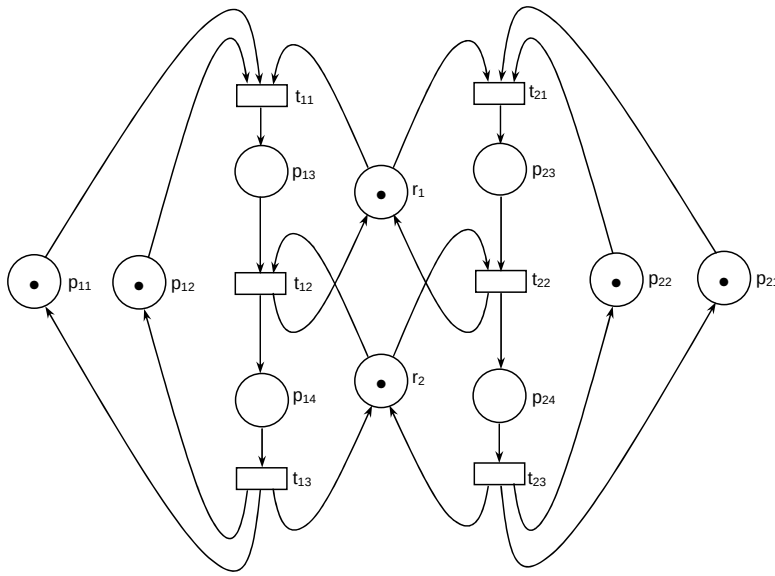

Fig. 15. The integrated system (Example 3)

6 CONCLUSION

Correctness and consistency are essential objectives in system design. The former requires that the system is free from erroneous situations, such as deadlock and capacity overflow. The latter requires that the system reflects exactly the functionalities of the given processes - no unrealistic processes and no unintended processes.

Typically, in a shared-resource system, as the resources are scarce, they used to be shared among different competing processes. If the system is not carefully designed, deadlocks originating from the competition of shared resources may occur. Besides, as resources have finite and limited capacity, the situation of capacity overflow must be avoided. Hence, design correctness, in terms of liveness, boundedness and reversibility, is an important issue in designing shared-resource systems. In this paper, after reviewing the properties of augmented marked graphs, we proposed a synthesis method for deriving a system design from a set of processes, especially for shared-resource systems.

The synthesis method has a number of distinctive features. It is based on augmented marked graphs whose structural characteristics are desirable for representing shared resources. The method begins with specifying a given set of processes as augmented marked graphs. These augmented marked graph are then synthesised into one single integrated net through the fusion of common places. The integrated net thus obtained serves to represent the integrated system which reflects exactly the functionalities of the given processes in the sense that the event sequences as well as the pre-conditions and post-conditions of each event occurrence are preserved. As the integrated net is structurally an augmented marked graph, the liveness, boundedness and reversibility of the system can be effectively analysed by making good use of the desirable properties of augmented marked graphs.

In principle, our method is generally applicable to the design of shared-resource systems. In this paper, we show a specific application to manufacturing system design. A manufacturing system is typically a shared-resource system, where resources used to be maximally shared among different competing processes. In manufacturing system design, a system design is derived from a given set of manufacturing processes. For a number of concerns pertaining to shared resources and capacity limits, it is essentially required to check if the system is live, bounded and reversible. Our synthesis method is applied to manufacturing system design, where the design correctness can be analysed effectively.

Acknowledgement

I would like to thank the editor and referees for their valuable and constructive comments on the first draft of this paper. Their comments are very useful for me to improve the paper in both the quality and presentation.

REFERENCES

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

[2] Smith, E.: On Net Systems Generated by Process Foldings. Advances in Petri Nets 1991, Lecture Notes in Computer Science, Vol. 524, pp. 253–295, Springer-Verlag, 1991.

[3] Hiraishi, K.: Construction of a Class of Safe Petri Nets by Presenting Firing Sequences. Application and Theory of Petri Nets 1992, Lecture Notes in Computer Science, Vol. 616, pp. 244–262, Springer-Verlag, 1992.

[4] Bordeleau, F.—Buhr, R. J. A.: 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 Press, 1997.

[5] Bordeleau, F.—Corriveau, J. P.—Selic, B.: A Scenario-Based Approach to Hierarchical State Machine Design. Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 78-85, IEEE Press, 2000.

[6] Chao, D. Y.—Zhou, M. C.—Wang, D. T.: Extending Knitting Technique to Petri Net Synthesis of Automated Manufacturing Systems. The Computer Journal, Vol. 37, 1994, No. 1, pp. 67–76.

[7] Chao, D. Y.—Wang, D. T.: Two Theoretical and Practical Aspects of Knitting Technique: Invariants and a New Class of Petri Net. IEEE Transactions on Systems, Man and Cybernetics, Vol. 27, 1997, No. 6, pp. 962–977.

[8] Jeng, M. D.—DiCesare, F.: Synthesis Using Resource Control Nets for Modeling Shared-Resource Systems. IEEE Transactions on Robotics and Automation, Vol. 11, 1995, No. 3, pp. 317–327.

[9] Jeng, M. D.: A Petri Net Synthesis Theory for Modeling Flexible Manufacturing Systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 27, 1997, No. 2, pp. 169–183.

[10] Chu, F.—Xie, X.: Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming. IEEE Transactions on Robotics and Automation, Vol. 13, 1997, No. 6, pp. 793–804.

[11] Cheung, K. S.: New Characterisation for Live and Reversible Augmented Marked Graph. Information Processing Letters, Vol. 92, 2004, No. 5, pp. 239–243.

[12] Cheung, K. S.—Chow, K. O.: Cycle Inclusion Property of Cycle-Inclusion Property of Augmented Marked Graphs. Information Processing Letters, Vol. 94, 2005, No. 6, pp. 271–276.

[13] Peterson, J. L.: Petri Net Theory and the Modeling of System. Prentice Hall, 1981.

[14] Reisig, W.: Petri Nets: An Introduction. Springer-Verlag.

[15] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, 1989, No. 4., pp. 541–580.

[16] Huang, H. J.—Jiao, L.—Cheung, T. Y.: Property-Preserving Composition of Augmented Marked Graphs That Shared Common Resources. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1446–1451, IEEE Press, 2003.

[17] Barkaoui, K.—Couvreur, J. M.—Dutheillet, C.: On Liveness in Extended Non Self-Controlling Nets. Application and Theory of Petri Nets 1995, pp. 25–44, Springer-Verlag, 1995.

[18] Chryssolouris, G.: Manufacturing Systems: Theory and Practice. Springer-Verlag, 1992.

[19] Dicesare, F.: Practice of Petri Nets in Manufacturing. Chapman and Hall, 1993.

[20] Descrochers, A. A.—Al-Jaar, R. Y.: Applications of Petri Nets in Manufacturing Systems. IEEE Press, 1994.

[21] Proth, J. M.—Xie, X.: Petri Nets: A Tool for Design and Management of Manufacturing Systems. Wiley, 1996.

[22] Sohdi, R. S. et. al. (Ed): Advances in Manufacturing Systems: Design, Modeling and Analysis. Elsevier, 1994.

[23] Wu, B.: Manufacturing Systems Design and Analysis. Chapman and Hall, 1994.
[24] Zhou, M. C.—Venkatesh, K.: Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach. World Scientific, 1999.

King Sing Cheung received his B. Sc. and Ph. D. in computer science from the City University of Hong Kong and his Master of Public Administration from the University of Hong Kong. He is a Chartered Engineer and a Chartered Scientist, and holds professional membership of the British Computer Society, the Institution of Mathematics and Its Applications, the Institution of Electrical Engineers and the Institute of Electrical and Electronic Engineers. He is currently an Information Technology Manager in the Hong Kong Baptist University. Prior to this, he was with the Chinese University of Hong Kong and the Open University of Hong Kong. His research interests include Petri net, formal specification and verification, object-oriented system design, use-case-driven system design, component-based software engineering, robotics and automation.