

Applications of Property-Preserving Algebras to Component-Based Manufacturing System Design*

HE-JIAO HUANG, TO-YAT CHEUNG AND XIAO-LONG WANG

*Department of Computer Science and Technology
Shenzhen Graduate School, Harbin Institute of Technology
Shenzhen 518055, China*

E-mail: hjhuang@hitsz.edu.cn, cscheung@alumni.cityu.edu.hk, wangxl@insun.hit.edu.cn

This paper presents the applications of several property-preserving Petri net process algebras (PPPA) to the specification and verification of manufacturing system design. It illustrates the applications by designing a Manufacturing System (MS) with component-based approach. PPPA handles the following three problems: 1) Integrating the primitive modules by using composite operators to create the final model; 2) Handling resource sharing by using place merging operators; and 3) Specifying the machining and/or assembly operations by using place refinement operators. Among other features, PPPA does not need to verify composite components because all the operators preserve many properties. Hence, if the primitive modules satisfy the desirable properties, each of the composite components, including the system itself, also satisfies these properties.

Keywords: component-based approach, manufacturing system design, Petri net, property-preserving algebra, verification

1. INTRODUCTION

Recently, researchers pay more attentions on the following features for manufacturing system (MS) design:

- *Resource sharing:* In a manufacturing system, resources such as robots, machines, assembly lines, buffers *etc.* sometimes are shared by several processes. Handling resource-sharing becomes a very important aspect during MS design. Note that ‘sharing’ does not imply simultaneous usage. While simultaneous usage can be handled by re-enterable codes in software systems and is not allowed in manufacturing systems, ‘sharing’ requires a resource to be occupied by some part(s) exclusively during utilization and is released afterwards. For multiple-resource systems, a wrong order in occupying and releasing these resources may cause deadlocks or overflow. In the literature, research for handling resource-sharing problems include mutual exclusion [29] and resource allocation [5, 23]. They assume that the resources will not be modified when they are switched from one process to another. In this paper, we handle the resource sharing problem by applying a property-preserving place-merging operator. Property-preserving means, if the original system satisfies some property, after the operators in the algebra were applied to it, the resultant system still has the same properties. Be-

Received December 17, 2004; revised March 2, 2005; accepted March 30, 2005.

Communicated by Ding-Zhu Du.

* This work was financially supported by the China Postdoctoral Science Foundation under grant No. 2005037180, and National Natural Science Foundation of China under grant No. 60435020.

sides the unchangeable case in the literature [13, 16, 29], we consider the case that the resources are modified when they leave one process for another.

- *Component-based architecture*: Component-based architecture allows implemented components to be constructed from some existing components. The architecture enlarges the reusability of the system modules and increases the system running speed. However, it faces many difficulties during integration, including operation system, interfaces, communication *etc.* In the literature, net-based approaches concerning integration include place merging [13, 22, 25], transition merging [26], path merging [2] and modular synthesis [8, 9, 28]. This paper handles the integration problem by applying property-preserving composition operators [20] such as enable, choice, interleaving and iteration, *etc.*
- *Property-preserving algebra*: In the manufacturing system design, when processes are modified or constituted for creating new processes, the designed properties usually be destroyed. In order to preserve the desirable properties, researchers pay many efforts on searching for the property-preserving operators for MS design. For example, [24] considered many property-preserving reduction rules, [10, 11, 12, 24] considered refinement methodologies and [13, 20, 22, 25] considered composition rules. [1, 3, 7, 20] include reduction, refinement and composition. Many more references can be found from the survey papers [4, 15]. Although there are many synthesis and reduction results for Petri nets, they need to be improved from the following perspectives: (a) The approaches are applied not only to special Petri nets, but also general Petri nets; (b) Many more system properties such as reversibility, proper termination, siphon, trap and so on should be preserved when the modules are merged and/or linked or stepwise refinements are done; and (c) Property-preserving should be verified in the approaches.

The Property-Preserving Petri Net Process Algebra (PPPA) [12, 16, 20, 24] applied in this paper overcomes the difficulties mentioned above.

The elements of PPPA are Petri net processes (PNP), a Petri net with a unique entry place, a unique exit place and a set of places for handling resource sharing. PNP is an extension of Workflow net [1] and agent net [19] by considering the static markings. PPPA has four types of operators: extensions, compositions, refinements and reductions. All operators can preserve about twenty properties (some under additional conditions), such as liveness, boundedness, reversibility, RC-property, traps, siphons, proper termination, *etc.* All the results concerning PPPA have been published or will be published in the literature [12-14, 16, 20, 24].

For designing the MS, our component-based approach starts by specifying the first level primitive modules of the MS as PNPs. Then, according to the control flow requirements of the system, the first level MS are created by integrating the primitive modules with applications of PPPA. Finally, the implementation level MS is obtained by applying refinements in PPPA. Among other features, this approach has two desirable features for the design of MS: (1) It eliminates the need of verifying the composite components by requiring all the operators to preserve the desirable properties. Hence, if the primitive PNPs satisfy these properties, each of the composite components, including the

MS itself, also satisfies these properties. (2) By using a property-preserving place-merging technique, it avoids such errors as deadlock and overflow that often arise when handling resource-sharing. The consideration about changeable resources makes the resource sharing problem more flexible.

It should be mentioned that, in the literature, Petri net based manufacturing systems design can be summarized into two basic approaches: bottom-up and top-down. Bottom-up approach begins with the primitive modules and the final system was constructed by merging and/or linking of all these modules. Top-down approach begins with the first level Petri net model, the implementation level was reached by the stepwise refinement for the first level model. The component-based approach applied in this paper is in fact a combination of bottom-up and top-down approaches. Hence, the results concerning both bottom-up [2, 8, 9, 13, 25, 26] and top-down approaches [10-12, 24] are also available in our approach.

For the rest of the paper, section 2 outlines how to apply PPPA to a component-based approach for MS design. Based on PPPA, the approach for specifying and verifying the design of MSs is illustrated in detail with an example. Section 3 is devoted to the specification of the primitive modules and the creation of the composite components of MS and the verification of these components. Some concluding remarks are given in section 4.

2. SUMMARY OF APPLYING PPPA TO COMPONENT-BASED APPROACH

Briefly speaking, our component-based approach is a combination of bottom-up and top-down approaches. It starts with the creation of the first level primitive modules of the system. Then, similar to bottom-up approach, the first level system model is constructed by combining the modules. Finally, the implementation model is obtained by stepwise refinement. PPPA is applied mainly to the last two steps for specification and verification. In the following, we describe how PPPA is applied to each step of the component-based approach.

Step 1: Creating the primitive modules.

The subsystems and resources are described as autonomous primitive modules. Each module is specified as a Petri net process (PNP) [20].

Note that for different systems, the modeling methods for the primitive modules may be different. For example, in a multi-agent system, the primitive modules such as message checker, resource-request handler, protocol selector *etc.* and the resources such as knowledge bases, environment, plan *etc.* are modeled according to their functions. In the workflow system, the primitive modules are modeled according to the workflow of the system. In this paper, the primitive modules for a manufacturing system are modeled according to the relations between the basic operations and the resources used in the operations in the system. In order to simplify the verification procedure and provide a correct system model, each operation of the system is modeled as a single place. Hence, the structure for each primitive module is very simple and the verification for each module is trivial.

Step 2: Creating the first level system model.

According to the relations between the primitive modules, the PNPs created in step 1 are integrated via some composition operators such as choice, enable, *etc.* or modified by applying some refinement and reduction operators. Resource sharing, if being part of the requirements, will be resolved by place merging. The resultant system is also a PNP.

In this step, PPPA are mainly applied to solve the following problems:

(1) Integrating the modules by using composition operators and reduction operators.

Previous research integrates the components by merging places [13, 22, 25], merging transitions [26], merging paths [2] or modular synthesis [8, 9, 28]. By the integration methods, except P -invariant, liveness and boundedness, no other properties can be verified. In this paper, composition operators such as choice, enable, *etc.* in PPPA are applied to the integration of components. To simplify the structure of the system models and do not change the functions of the system, reduction operators are used. Composition operators and reduction operators can preserve not only liveness, boundedness and reversibility. They can preserve or conditionally preserve many other properties such as proper termination, P -invariant, siphon, trap and so on.

(2) Handling resource sharing by place merging.

Handling resource sharing is a hard problem in the system design. [29] solves this problem by using parallel and sequential exclusion techniques. Although it is an efficient tool for verifying liveness, boundedness and reversibility, the technique itself is too complex and cannot be used to check many other properties of the system.

Place merging is found to be very efficient for handling resource sharing problems in systems design. In the literature, conclusions for place merging are limited to preserving P -invariant [13], T -invariant [22] and liveness [25]. Recently, preserving many other properties is proved possible for place merging operators [16].

In this paper, the resources used in different modules are considered to be different. As will be seen in section 3, these resource places representing the same resources are merged into single ones.

Step 3: Creating the implementation level system model.

In step 1, each module is modeled as the first level model. Hence, the model for the system after integrating the components is still at the first level. In order to obtain the implementation model, some places or transitions are needed to be refined with their detailed functional models. In this paper, since each operation is modeled as a single place, place refinement is applied to specify the operations. In the previous research, resources are considered to be unchangeable during proceeding in the system. In this paper, resources are permitted to be modified when they leave one process for another. This assumption makes our models more flexible. Hence, after place merging for handling resources sharing, place refinement should be applied to the resource places in order to specify the modification for the resources.

In fact, this step is similar to the top-down approach. Hence, besides place refinement operators in this paper, other synthesis and reduction techniques [10-12, 24] in the top-down approaches are applicable in this step when necessary.

Step 4: Verifying the system model.

No much effort is needed for verification in our approach because of the property-preserving characteristics of all the operators. In fact, if the constituent PNPs satisfy such properties as liveness, boundedness, reversibility and proper termination, the intermediate PNPs obtained by applying these operators satisfy these properties as well. However, the burden is shifted to making sure that the initial PNPs satisfy these properties.

3. APPLICATION OF PPPA TO MANUFACTURING SYSTEM DESIGN

This section illustrates the application of PPPA with an example of designing a manufacturing system. The organization of this section is according to the steps described in section 2.

3.1 Creating the Primitive Modules

This subsection specifies the various primitive modules of the manufacturing system as a set of PNPs, which corresponds to step 1 outlined in section 2.

The manufacturing system constructs the final product from three primitive parts by using four machining centers MC_i (each MC_i contains a machine M_i), $i = 1, 2, 3, 4$, two assembly stations A_1 and A_2 , two robots R_1 and R_2 and a buffer B .

The production is produced as follows:

1. Part 1 is machined by M_1 . Part 2 is machined by M_2 . Each part is automatically fixtured to a pallet and loaded into the machine.
2. After processing, Parts 1 and 2 enter assembly station A_1 for producing Part S . When either A_1 or A_2 is ready to execute the assembly task, it requests both robots R_1 and R_2 and acquires them if they are available. When A_1 (A_2) completes, it releases both robots.
3. Part 3 is machined first by M_3 and then by M_4 . In M_3 , the part is automatically fixtured to a pallet and loaded into the machine. After processing, robot R_1 unloads the intermediate part from M_1 into buffer B and M_1 is released.
4. At machining center MC_4 , intermediate part is automatically loaded into M_4 and processed. When M_4 finishes processing a part, robot R_1 unloads the final product Part T and releases M_4 .
5. Assembly station A_2 assembles Parts S and T to produce the final products.
6. It is assumed that input parts are always available to be fixtured and that the finished products are removed.
7. Robot R_1 needs to be modified when it is switched from one user to another. When R_1 is switched from A_1 to A_2 , it needs to be oiled; before entering M_3 or M_4 from A_2 , robot R_1 needs the addition of some parts; before R_1 is switched from M_3 or M_4 to A_1 , the oil left from M_3 or M_4 should be cleaned.

We assume that once the system is executed, it cannot be interrupted. To avoid mixing two independent execution cycles of a PNP, one has to either use colored Petri nets or control the procedure of entering into the process. We adopt the second approach in

order to simplify the illustration. In other words, the system cannot begin a new iteration before termination.

• Modeling the Primitive Modules in the Manufacturing System (Fig. 1).

In our approach, each module is specified as a marked process. When the entry place is marked, the process is executable. When the exit place has a token, the process terminates immediately. A place represents a resource status or an operation. A transition represents either start or completion of an operation process.

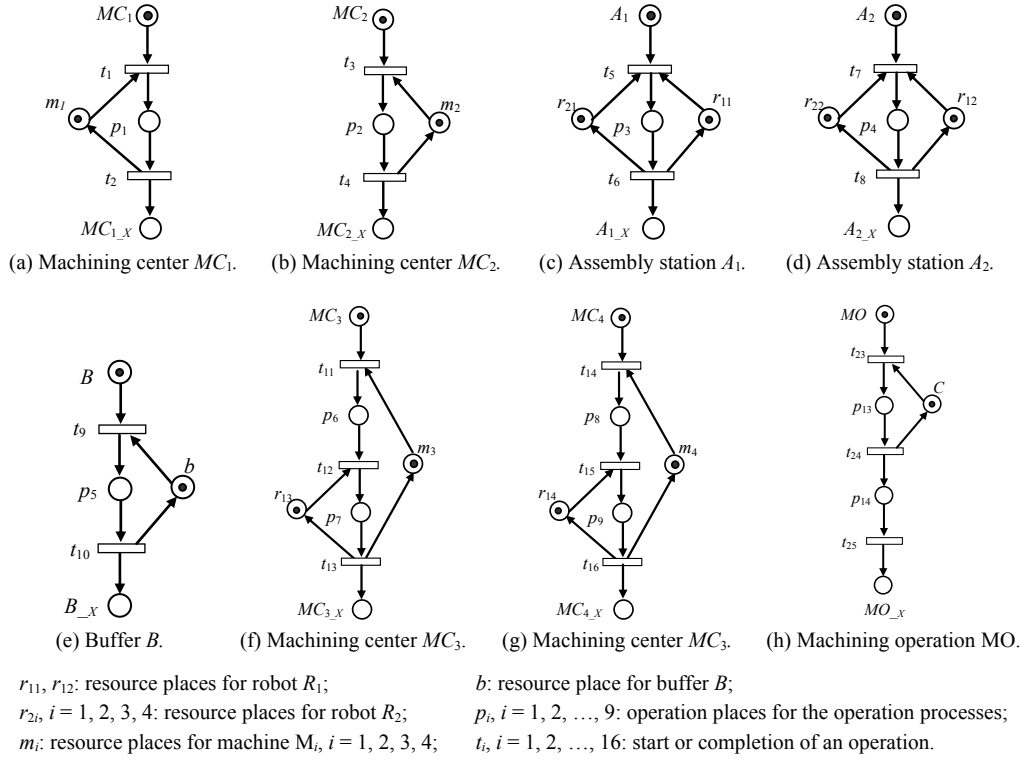


Fig. 1. Petri net processes for the primitive modules of MS.

Machining Center MC_1 (Fig. 1 (a)): This module has one operation for machining Part 1. When the machine M_1 is available (*i.e.*, resource place m_1 has a token), Part 1 begins to be machined by firing transition t_1 . p_1 is the operation place, when it has a token, the machining operation is performed. Transition t_2 represents the completion of the machining operation. After firing t_2 , the machine M_1 is available again.

Machining Center MC_2 (Fig. 1 (b)): The Petri net process for this module is similar to process MC_1 except that it is for machining Part 2.

Assembly Station A_1 (Fig. 1 (c)): This module assembles Parts 1 and 2 to produce Part S . When both robots are available (*i.e.*, resource places r_{11} and r_{21} have tokens), the assem-

bly can begin by firing transition t_5 . When the assembly is completed, *i.e.*, firing transition t_6 , both robots are released.

Machining Center MC_3 (Fig. 1 (f)): This module has two operations, *i.e.*, machining Part 3 and unloading the intermediate part. When the machine M_3 is available, *i.e.*, place m_3 has tokens, Part 3 can be machined by firing transition t_{11} . Place p_6 is the operation place, representing performing of the machining operation when it has tokens. After machining, the intermediate part is unloaded if robot R_1 is available. Transition t_{12} represents the completion of machining operation and the beginning of the unloading operation. Place p_7 is the operation place, representing the process of unloading operation. After unloading, both robot R_1 and machine M_3 are released.

Buffer B (Fig. 1 (e)): This module stores the intermediate part produced from module MC_3 . When the buffer is available (*i.e.*, resource place b has a token), the intermediate part can be stored by firing transition t_9 . When the intermediate part is removed by firing transition t_{10} , the buffer is released.

Machining Center MC_4 (Fig. 1 (g)): The intermediate part produced by MC_3 and stored in buffer B is processed in this module to produce Part T . The process is similar to that in module MC_3 and thus the Petri net process for this module is similar to MC_3 .

Assembly Station A_2 (Fig. 1 (d)): Parts S and T are assembled in this module. The process is similar to that in module A_1 and thus the Petri net process for this module is similar to A_1 .

Machining Operation MO (Fig. 1 (h)): This is a Petri net process specifying the operations in Machining Center MC_1 . When the conveyor C is available, raw materials for Part 1 are moved from the storage to the Machining Center MC_1 . Transitions t_{23} and t_{24} represent the start and completion of the movement operation, respectively. Place p_{13} is the operation place. After the materials enter the machine, Part 1 begins to be machined. Place p_{14} represents the machining process and its associated transitions represent the start and completion of the machining process, respectively.

3.2 Creating the System Model

This subsection describes in detail how to apply the operators of PPPA for creating MS from the PNPs obtained in section 3.1., which corresponds to steps 2-3 outlined in section 2.

Step 2.1: Integrating the modules by using composition and reduction.

This sub-step applies the composition operators and reduction operators on the PNPs of Fig. 1. The intermediate PNPs are shown in Fig. 2. The technical details about the composition operators ENABLE, CHOICE, INTERLEAVING can be found in [20] and the reduction operator REDUCE-PATH can be found in [20].

- (a) Modules MC_1 and MC_2 are integrated with the operator INTERLEAVING, resulting in the PNP MC_{12} (Fig. 2 (a)). This integration is motivated by the fact that MC_1 and MC_2 handle Parts 1 and 2 asynchronously.

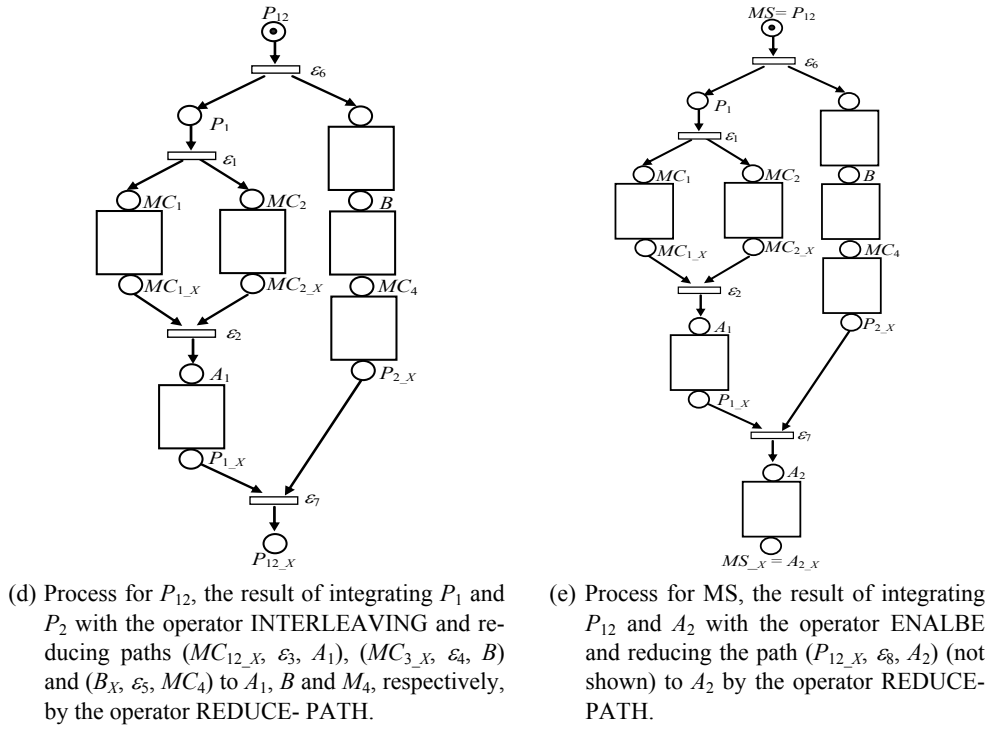
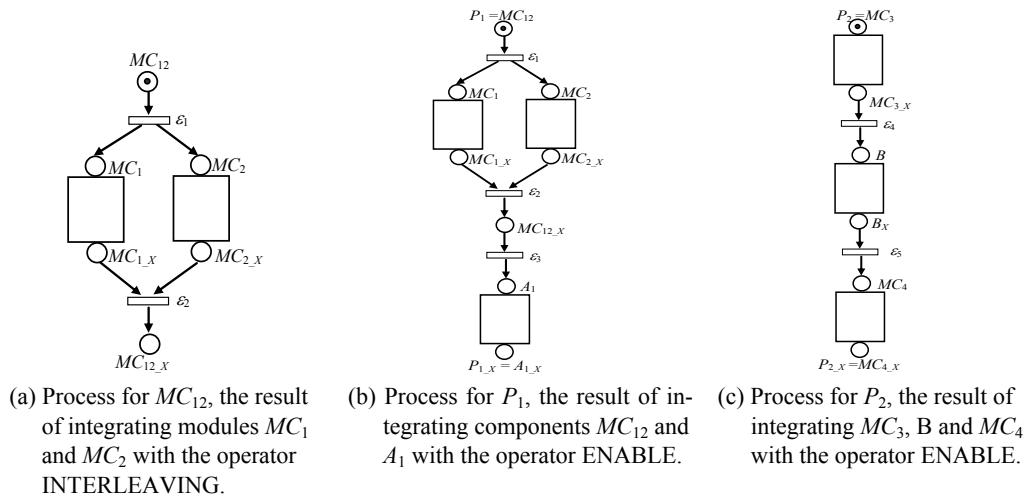


Fig. 2. Step 2.1 for constructing the manufacturing system MS .

(b) Components MC_{12} and A_1 are integrated with the operator ENABLE, resulting in the PNP P_1 (Fig. 2 (b)). This integration is motivated by the fact that, after machining Parts 1 and 2 in MC_1 and MC_2 , respectively, the next step is to assemble these parts in A_1 .

- (c) Modules MC_3 , B and MC_4 are integrated with the operator ENABLE, resulting in the process P_2 (Fig. 2 (c)). This integration is motivated by the fact that, Part 3 is first machined in M_3 , then stored in the buffer B and at last machined in M_4 .
- (d) Components P_1 and P_2 obtained in (b) and (c), respectively, are first integrated with the operator INTERLEAVING. Then, the paths $(MC_{12_X}, \varepsilon_3, A_1)$, $(MC_{3_X}, \varepsilon_4, B)$ and $(B_X, \varepsilon_5, MC_4)$ are reduced to single places A_1 , B and MC_4 , respectively, by using the operator REDUCE-PATH. The result is the PNP P_{12} shown in Fig. 2 (d). This integration is motivated by the fact that P_1 and P_2 handle Parts S and T asynchronously.
- (e) Components P_{12} and A_2 are first integrated with the operator ENALBE. Then, the path $(P_{12_X}, \varepsilon_8, A_2)$ (not shown) is reduced to place A_2 by using the operator REDUCE-PATH. The result is the PNP MS shown in Fig. 2 (e). The Petri net representation of process MS is shown in Fig. 3 (a).

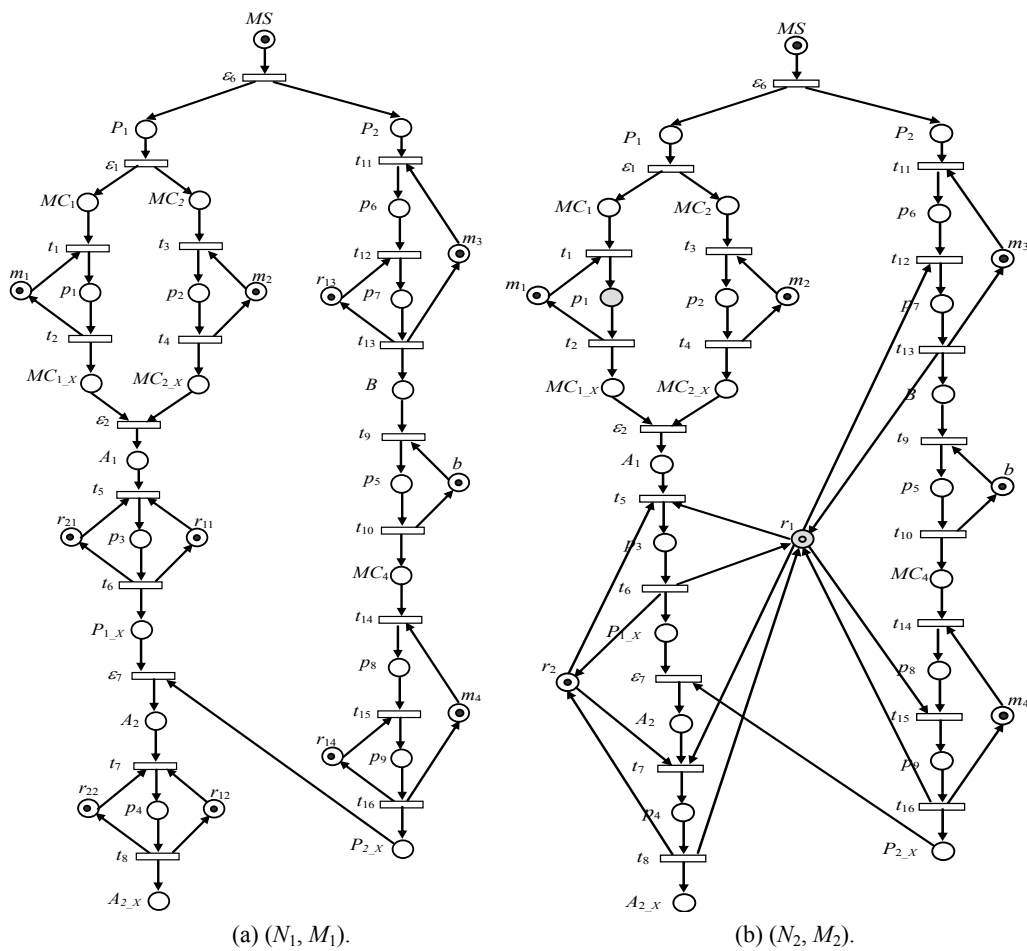


Fig. 3. Petri net Processes for MS before and after merging the resource places.

Step 2.2: Handling resource sharing by using place merging.

This step resolves the resource sharing problem arising in step 2.1. The shared robots R_1 and R_2 are represented in Fig. 3 by places having the same label but with different suffices. For example, robot R_1 , the resource shared by A_1 , A_2 , MC_3 and MC_4 , has place labels r_{11} , r_{12} , r_{13} and r_{14} , respectively. Robot R_2 , the resource shared by A_1 and A_2 , has place labels r_{21} and r_{22} , respectively. Obviously, in an integrated system, the places representing the same resource should be merged into a single place. Such merging is done according to REDUCE-N-PLACE [16]. The resulting PNP is shown in Fig. 3 (b).

Step 3: Creating the implementation level system model by stepwise refinement.

In our modeling approach, each operation is represented by an operation place. In order to provide a more detailed specification of the operation processes, the operation places can be refined with their detailed Petri net processes. The PNP Machining Operation MO in Fig. 1 (h) is an example for the operation place p_1 in the module MC_1 (Fig. 1 (a)) and the process MS (Fig. 3 (b)). The refinement process is done by using the operator PLACE-REFIENMENT [12, 24]. To simplify the appearance of the figure, refining place p_1 in Fig. 3 (b) is not shown.

The ‘place merging’ process in step 2.2 shows that robot R_1 is shared by several processes. However, R_1 is assumed to be able to accommodate the shift differences between those modules. Hence, place r_1 in Fig. 3 (b) represents a function call to a subnet called Robot Modification RM shown in the ellipse in Fig. 4. This subnet handles three operations, *i.e.*, oiling, adding parts and cleaning oil. Places p_{10} , p_{11} and p_{12} represent these three operations, respectively. Their associated transitions represent the start or completion of the corresponding operations, respectively. After refinement for Fig. 3 (b), the resulting PNP is shown in Fig. 4.

3.3 Verifying the System Model

In this subsection, it will be shown that all the primitive modules created in step 1 and all the composite components created in steps 2 and 3 are correct, which corresponds to step 4 outlined in section 2. For this example, a component is considered to be *correct* if it is almost live, bounded and almost reversible and terminates properly. In general, it is up to the designer to select the set of properties for defining the meaning of correctness.

- Correctness of the primitive modules created in section 3.1:

A component-based approach starts with a set of correct primitive modules. For MS, they are PNPs MC_1 , MC_2 , MC_3 , MC_4 , A_1 , A_2 , B and MO as shown in Fig. 1. Since their structures are quite simple, it is not hard to see that every one of them is a PNP and is correct. For example, all of them cannot self-start. Since every associated process in Fig. 1 is a strongly connected marked graph and each cycle has exactly one token, the PNPs in Fig. 1 are almost live and bounded [21]. Another example is the proper termination property of the module MC_3 in Fig. 1 (f). This PNP has only one possible firing sequence $\sigma = t_{11}t_{12}t_{13}$ such that, after firing the sequence, a token is deposited into the exit place $MC_{3,X}$ and, by the definition of a process, execution should terminate. At this moment, initial token distribution (except that the token in the entry place is now in the exit place) is resumed, meaning that termination is proper.

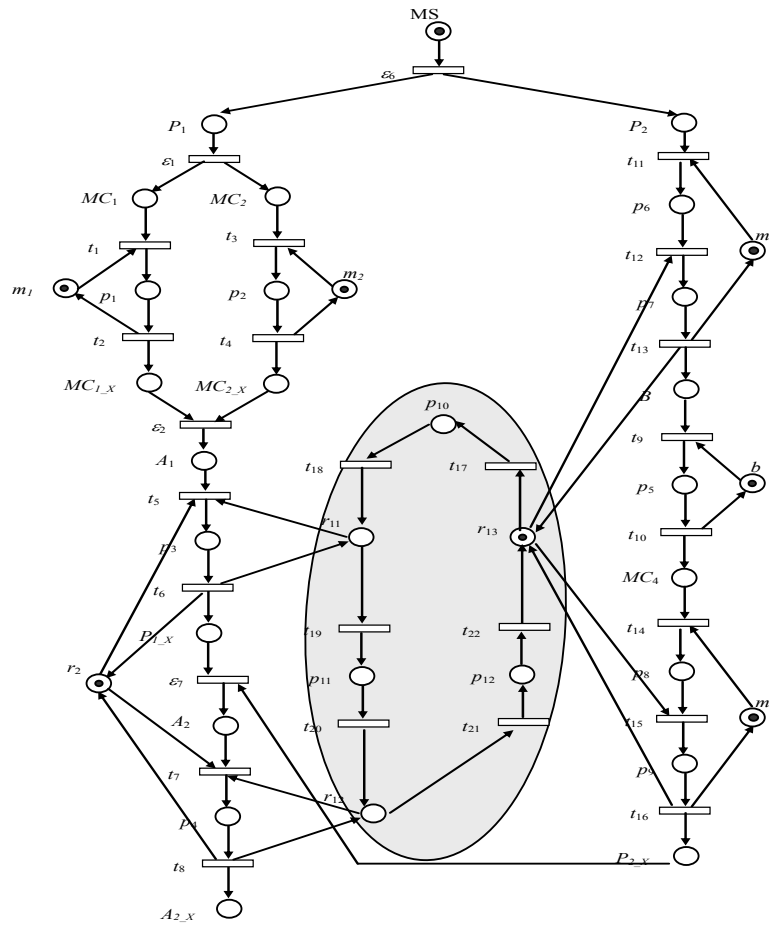


Fig. 4. Process MS after applying the place refinements.

- Correctness of the PNPs after composition and reduction in step 2.1 of section 3.2:

It will be shown that every PNP produced in step 2.1 of section 3.2 is correct. In step 2.1, MC_1 and MC_2 are first integrated by using the operator INTERLEAVING, resulting in MC_{12} (Fig. 2 (a)). Next, MC_{12} and A_1 are integrated by using the operator ENABLE, resulting in P_1 (Fig. 2 (b)). MC_3 , B and MC_4 are integrated by using the operator ENABLE again, resulting in P_2 (Fig. 1 (c)). Then, components P_1 and P_2 are integrated by using INTERLEAVING again, resulting in P_{12} (Fig. 2 (d)). Since MC_1 , MC_2 , A_1 , MC_3 , B and MC_4 are all correct, it follows from Theorems 6.1, 6.2 and 6.3 in [20] that P_{12} is correct. Step 2.1 (d) also reduces the paths $(MC_{12,X}, \varepsilon_3, A_1)$, $(MC_{3,X}, \varepsilon_4, B)$ and $(B_X, \varepsilon_5, MC_4)$ to places A_1 , B and MC_4 , respectively. Since $\bullet MC_{12,X} \neq \emptyset$, $\bullet MC_{12,X}^\bullet = \{\varepsilon_3\}$ and $M_0(A_1) = 0$, $\bullet MC_{3,X} \neq \emptyset$, $\bullet MC_{3,X}^\bullet = \{\varepsilon_4\}$ and $M_0(B) = 0$, $\bullet B_X \neq \emptyset$, $\bullet B_X^\bullet = \{\varepsilon_5\}$ and $M_0(MC_4) = 0$, by Theorems 3.4 and 4.2 in [12], the properties are preserved. Hence, P_{12} is correct. In step 2.1 (e), P_{12} and A_2 are integrated by using the operator ENABLE and the path $(P_{12,X}, \varepsilon_8, A_2)$ is reduced to place A_2 . Since both P_{12} and A_2 are correct, by Theo-

rem 6.1 in [20], the produced process MS_0 (not shown) obtained by integrating P_{12} and A_2 is correct. In MS_0 , since $A_2^* \neq \emptyset$, $^*A_2 = \{\varepsilon_8\}$ and $M_0(A_2) = 0$, the condition of Theorem 2 in [14] is satisfied. Hence, reducing the path preserves the correctness and the process MS is correct. That is, the Petri net process (N_1, M_1) for MS shown in Fig. 3 (a) is almost live, bounded, almost reversible and terminates properly.

- Correctness of the PNPs after place merging in step 2.2 of section 3.2:

In step 2.2, two sets of resource places $Q_1 = \{r_{11}, r_{12}, r_{13}, r_{14}\}$ and $Q_2 = \{r_{21}, r_{22}\}$ in (N_1, M_1) (Fig. 3 (a)) are merged by REDUCE-N-PLACE to r_1 and r_2 , respectively, resulting in Petri net process (N_2, M_2) (Fig. 3 (b)) and $P_0 = \{MS, P_1, P_2, MC_1, MC_2, MC_4, m_1, m_2, m_3, m_4, MC_{1_X}, MC_{2_X}, A_1, P_{1_X}, P_{2_X}, A_2, A_{2_X}, B, b, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$. In the following, it will be shown that (N_2, M_2) is almost live, bounded and almost reversible by applying results in [16]. Furthermore, it can be observed that (N_2, M_2) terminates properly.

We can verify the following conditions are satisfied in (N_1, M_1) : (1) (N_1, M_1) satisfies the rsd-MST-property [16]; (2) For every $p \in P_0$, there exists an rsd-siphon D such that $p \in D$ and the D -induced subnet N_D is a state machine; (3) For every Q_i where $i=1, 2$, there exists an rsd-siphon D such that $Q_i \subseteq D$ and the D -induced subnet N_D is a state machine. By the results in [16], (N_2, M_2) is almost live and bounded.

In the following, we will show that the Petri net (N_2, M_2) is almost reversible. Firstly, it will be shown that the associate process of (N_1, M_1) is a live augmented marked graph by checking the conditions in the definition of augmented marked graph [6, 30] one by one. 1) $(N_1, M_1) | (Q_1 \cup Q_2)$ is almost live and bounded. After deleting some resource places, the correctness of each process in Fig. 1 is not changed. By step 2.1, (N_1, M_1) is still correct even if the resource places are deleted. 2) Each resource place is associated with a transition pair (a_i, b_i) and there exists an elementary path O_i from a_i to b_i . For example, resource place r_{11} is associated with (t_5, t_6) and there exists an elementary path (t_5, p_3, t_6) from t_5 to t_6 . 3) Every resource place is marked by M_1 and no elementary path O_i is marked by M_1 . Secondly, for every siphon D of N_1 , if $Q_i \subseteq D$, there exists a trap S marked by M_1 within D such that $S \subseteq P_0 \cap D$ or $Q_i \subseteq S$. By [16], (N_2, M_2) is almost reversible.

Finally, it can be observed that (N_2, M_2) terminates properly.

- Correctness of the PNPs after the refinements in step 3 of section 3.2:

Since process MO is correct and place p_1 in MC_1 (therefore in the process (N, M_0)) has input transition t_1 , it follows from Theorems 3.4 and 4.2 in [12] that refining place p_1 with the process MO shown in Fig. 1 (h) preserves the correctness. For simplicity, refining place p_1 in (N, M_0) is not shown.

Since refinement subnet RM is a strongly connected state machine, by 3 in [14], refining place r_1 with subnet RM preserves the correctness of (N_2, M_2) . Hence, the Petri net process (N, M_0) in Fig. 4 is correct.

4. CONCLUSION

Due to its great variation in requirements and proneness to modification, it is not feasible to pinpoint a specific architecture and an initial set of primitive modules for the

design of general MS. It seems more appropriate just to adopt an open, correctness-guaranteed and component-based approach so that different MSs may start from different primitive modules and be composed in different ways. This paper has presented a formal approach for such a purpose. The approach has two main features: (1) It includes an algebra that provides the means for representing the MS modules as Petri net processes, integrating simple modules into composite ones. Though only three composition operators, namely, ENABLE, CHOICE, and INTERLEAVING, one refinement operator and some reduction operators are described in this paper because of space limitation, this algebra in fact includes many other operators [1, 3, 7, 10, 11, 17, 18, 20, 27]. Hence, very complex modules and MS architectures can be created. (2) A property-preservation requirement is embedded into the algebra for guaranteeing the correctness in every step of the integration process. In fact, every operator of this algebra can preserve about twenty properties, though only the four most basic ones, namely, liveness, boundedness, reversibility and proper termination are considered in this paper. Hence, manufacturing system with a great variety of property requirements can be designed.

REFERENCES

1. W. V. D Aalst, "Verification of workflow nets," *Lecture Notes in Computer Science*, Vol. 1248, 1997, pp. 407-426.
2. J. S. Ahuja and K. P. Valavanis, "A hierarchical modeling methodology for flexible manufacturing systems using extended Petri nets," in *Proceedings of the International Conference on Computing Integrated Manufacture*, 1988, pp. 350-356.
3. E. Best, R. Devillers, and M. Koutny, *Petri Net Algebra*, Springer, 2001.
4. W. Brauer, R. Gold, and W. Vogler, "A survey of behavior and equivalence preserving refinement of Petri nets," *Lecture Notes in Computer Science*, Vol. 483, 1990, pp. 1-46.
5. J. M. Colom, "The resource allocation problem in flexible manufacturing systems," *Lecture Notes in Computer Science*, Vol. 2679, 2003, pp. 23-35.
6. F. Chu and X. Xie, "Deadlock analysis of Petri nets using siphon and mathematical programming," *IEEE Transactions on Robotics and Automation*, Vol. 13, 1997, pp. 793-804.
7. T. Y. Cheung and W. Zeng, "Invariant-preserving transformations for the verification of place/transition systems," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, 1998, pp. 114-121.
8. A. Datta and S. Ghosh, "Synthesis of a class of deadlock-free Petri nets," *Journal of ACM*, Vol. 31, 1984, pp. 486-506.
9. G. Goos and J. Hartmanis, "Modular synthesis of deadlock-free control structures," *Foundations of Software Technology and Theoretical Computer Science*, Vol. 241, 1986, pp. 288-318.
10. C. Girault and R. Valk, *Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications*, Springer, 2003.
11. N. Hamerlain, "Refinement of open protocols for modeling and analysis of complex interactions in multi-agent systems," *Lecture Notes in Artificial Intelligence*, Vol. 2691, 2003, pp. 423-434.

12. H. J. Huang, T. Y. Cheung, and W. M. Mak, "Structure and behavior preservation by Petri-net-based refinements in system design," *Theoretical Computer Science*, Vol. 328, 2004, pp. 245-269.
13. H. J. Huang, L. Jiao, and T. Y. Cheung, "Property-preserving composition of augmented marked graphs that share common resources," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003, pp. 1446-1451.
14. H. J. Huang, L. Jiao, and T. Y. Cheung, "Property-preserving subnet reductions for designing manufacturing systems with shared resources," *Theoretical Computer Science*, Vol. 332, 2005, pp. 461-485.
15. M. D. Jeng and F. DeCesare, "A review of synthesis techniques for Petri nets with application to automated manufacturing systems," *IEEE Transactions on System, Man, and Cybernetics*, Vol. 23, 1993, pp. 301-312.
16. L. Jiao, H. J. Huang, and T. Y. Cheung, "Handling resource sharing by property-preserving reductions in Petri-net based system design," submitted to *IEEE Transactions on Systems, Man and Cybernetics*.
17. M. Kohler, D. Moldt, and H. Rolke, "Modeling the structure and behavior of Petri net agents," *Lecture Notes in Computer Science*, Vol. 2075, 2001, pp. 224-241.
18. M. Kohler, D. Moldt, and H. Rolke, "Modeling mobility and mobile agents using nets within nets," *Lecture Notes in Computer Science*, Vol. 2679, 2003, pp. 121-139.
19. S. Ling and S. W. Loke, "MIP-nets: a compositional model of multi-agent interaction," *Lecture Notes in Artificial Intelligence*, Vol. 2691, 2003, pp. 61-72.
20. W. M. Mak, "Verifying property preservation for component-based software systems (a Petri-net based methodology)," Ph.D. Thesis, Dept. of Computer Science, City University of Hong Kong, H.K., 2001.
21. T. Murata, "Petri nets: properties, analysis, and applications," in *Proceedings of the IEEE*, Vol. 77, 1985, pp. 541-580.
22. Y. Narahari and N. Viswanadham, "A Petri net approach to the modeling and analysis of flexible manufacturing systems," *Annals of Operations Research* 3, 1985, pp. 449-472.
23. S. A. Reveliotis, "Accommodating FMS operational contingencies through routing flexibility," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998, pp. 573-579.
24. I. Suzuki and T. Murata, "A method for stepwise refinement and abstraction of Petri nets," *Journal of Computer and System Sciences*, Vol. 27, 1983, pp. 51-76.
25. Y. Souissi, "On liveness preservation by composition of nets via a set of places," *Lecture Notes in Computer Science*, Vol. 524, 1990, pp. 277-295.
26. I. L. Villarroel, J. Martinez, and M. Silva, "GRAMAN: a graphic system for manufacturing system design," in *Proceedings of the IMACS International Symposium on Systems Model and Simulation*, 1989, pp. 311-316.
27. H. Xu and S. M. Shatz, "A framework for model-based design of agent-oriented software," *IEEE Transactions on Software Engineering*, Vol. 29, 2003, pp. 15-30.
28. M. Zhou, F. DiCesare, and A. A. Desrochers, "A top-down modular approach to synthesis of Petri net models for manufacturing systems," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, pp. 534-539.
29. M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, 1993.

30. M. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: a Petri Net Approach*, World Scientific Publishing Co. Pte. Ltd, 1999.



He-Jiao Huang (黃荷姣) received her Ph.D. degree in Computer Science from City University of Hong Kong (CityU) in 2004, and her M.S. and B.S. degrees in Mathematics from Shaan Xi Normal University in 1999 and 1996, respectively. She had been a research assistant in CityU from March 1999 to January 2001. She is currently an Associate Professor in Harbin Institute of Technology Shenzhen Graduate School. Her research interest includes system design, information retrieval, Petri net theory and application, artificial intelligence, communication networks and so on.



To-Yat Cheung (張道一) received the B.S. degree in Mathematics from the University of Hong Kong, the M.S. and PhD degrees in Computer Science from the University of Wisconsin in 1962, 1967, and 1970, respectively. He has been a professor in the University of Ottawa and a chair professor in City University of Hong Kong for ten years, respectively. Currently, he is a visiting professor of Harbin Institute of Technology Shenzhen Graduate School. His research interest includes Distributed computing, Workflow, system design, Petri nets, etc.



Xiao-Long Wang (王曉龍) received the B.E. degree in Computer Science from the Harbin Institute of Electrical Technology, China, the M.E. degree in Computer Architecture from Tianjin University, China, and the Ph.D. degree in Computer Science and Engineering from Harbin Institute of Technology in 1982, 1984, and 1989 respectively. Currently, he is a Professor of Computer Science at Harbin Institute of Technology. His research interest includes artificial intelligence, machine learning, computational linguistics, and Chinese information processing.