Process-Translatable Petri Nets for the Rapid Prototyping of Workflow Processes

*Peter Pi-Te Chen *Chen-Chau Yang **Kuang-Lung Lin
*Department of Electronic Engineering
National Taiwan University of Science Technolog
Pete@rs590.ndmc.edu.t
**Graduate School of Decision
National Defense Management College

Abstract

This paper presents a method for the rapid prototyping of workflow process design usin g WPT-nets. The OPS83 rule-based language has a similar executive strategy to WPT-nets model. Hence, it may be simulated to support following activities: rapid prototyping, simulation, and automatic translation into program structures. In particular, WPT-nets are shown to be translatable into OPS83 rule-based program structures to allow users quickly develop simulation models of the workflow processes.

Keywords: 1. Petri Nets 2. Rapid Prototyping 3. Simulation 4. Workflo

I. Introduction

Some software dev elopments and increasing number of being built systems can not meet users actual needs. Current research efforts concentrate on developing a complete method-ology with integrated support tools to enhance the software quality, and reduce the software cost. The rapid prototyping is a promising method for easily specifying user requirement, requiring less system developing time, and its ease of refining syste specification.

Petri nets[9] are abstract formal models of information flow characterized by controls and constraints. It is now being widely applied to rapid prototyping[5], dataflow graphs[11], concurrent system analysis[12] and system performances[10].

This paper presents a method based on an extension of classical Petri nets for representing the specifications of workflow processes. The user requirements are modelled into Workflow Process Translation nets(shortly WPT-nets), and then translate these nets into production rules by using BNF. A rule parser is implemented for checking the syntax of the rules. If the parse find out syntax errors, the prod-uction rules can be refined again. Finally, A skeleton is presented to translate the WPT-nets into OPS83 rule-base program struc -tures that can perform automa-tically. The executed results can enhance the functional validation of user requirements and rapid prototyping integrity. The rapid proto-typing based on the WPT-nets shown in Figure 1. It supports the following activities:

(1) Specification: To translate the WPT-nets into rules by using BNF graph and use them as input specifications.
(2) Validation: WPT-nets can be analyized to determine properties such as deadlock, boundness, reachability, and conservation.
(3) Evaluation: WPT-nets are the extension of Petri nets. In WPT-nets, the deterministic timing is associated with transitions and can be used to discover critical aspects.
(4) Prototyping: A rule -based program skeleton[4][6][7][8] can be derived from the WPT-nets described by production rules, so that programmers can easily represent parallel workflo processes. The functional validation can check out the prototype program structure errors, and misunderstanding the specifications of the prototyping. Finally, a successfu prototyping can also be derived from the WPT-nets[13] if the target language is used

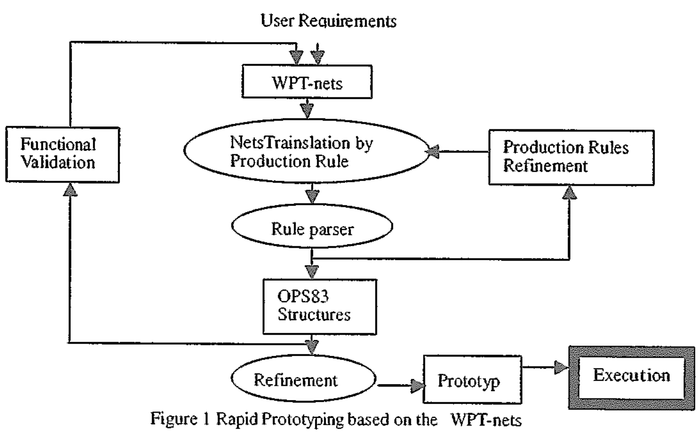


Figure 1 Rapid Prototyping based on the WPT-nets

to support the parallel process applications[1].

(5) Execution: The rapid prototyping ensures the correctness of the above input rules through the parser and automatically produces rule-based program codes.

The paper is organized as follows. Section II begins with model definitions. In this section, some properties of the WPT-nets are discussed. Section III presents translation of WPT-nets into OPS83 rule -base program structures. Section IV presents the implementation of the generator using our developed approach. Section V ,discusses several problems and strategies regarding the refinement of system specifications. Finally, section VI, summarized the result and its recommends for the future researches.

II. WPT-nets

The WPT-nets are composed of seven parts: a set of places P, a set of transitions T, an input function I, an output function O, a inhibitor arc mechanism i^0, and a marked time τ .

<Definition > A WPT-nets is a 7 tuple

$$WPT = <P, T, I, O, i^0, \mu^0, \tau>$$

where

$P = \{ p_1, p_2, p_3, \ldots p_n \}$ a set of places, $\geqq 0$

$T = \{ t_1, t_2, t_3, \ldots t_m \}$ a set of transitions, $m \geqq 0$

$I : T \rightarrow P^\infty$ is the input function

$O : T \rightarrow P^\infty$ is the output function

i^0 : is a inhibitor arc mechanism for "zero testing" a place.

μ^0 : is a marking on WPT, termed the initial marking.

τ : is a function $\tau : T \rightarrow \{1, 2, \ldots\}$, attached each transition in the net into one of the natural numbers. $\tau = (\tau_1, \ldots, \tau_m)$ in which $m = | T |$ and each $\tau (t_i) = \tau_1$

In the WPT-nets, static construction part consists of place and transition nodes. Place expresses states of the co rresponding process types and transition expresses synchronization among processes. Place and transition are connected by arcs, forming the directed graph. The dynamic construction part represents the moving of the token distributed on the places. Each place includes one or more than one token, but sometimes there are empty.

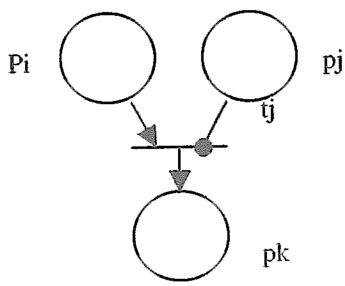


Figure 2.1 The inhibitor arc extension

The token distribution on the places is calling marking. Suppose there is at least one token existing in the input place of transition, then the transition as being enabled and fired.

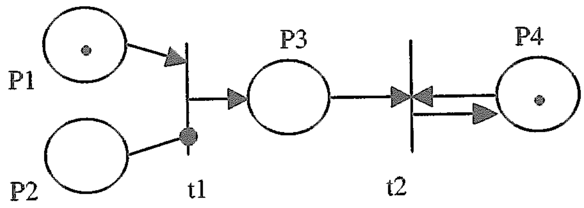


Figure 2.2 A marked WPT-net

Figure 2.3 represents the WPT-net after firing of transition t $_1$. A consequence of firing transition t_1 is that the transition may not be enable again because there is no token present in place $_1$. Another result of firing t_1 is that transition t_2 is now enabled, as there is now at least one token in each of the input place. In the WPT-nets, the deterministic timing is associated with a transition containing enabling time, firing time, and unavailable time [12]. The timing of the WPT-net is shown in Figure 2.4.

III . Translate the WPT-nets into OPS83

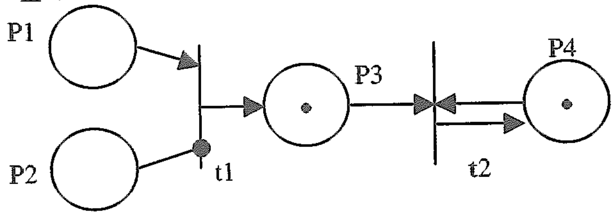


Figure 2.3 WPT-net after firing of transition t_1

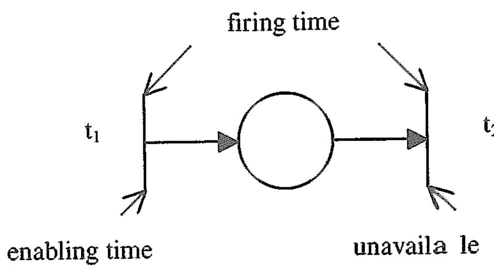


Figure 2.4 Timing of WPT-nets

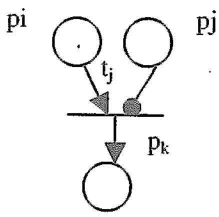
rule-base program structures

This section presents a high-level perspective of the system design consi sting of user level, kernel level, and user destination level. Examples of translating the WPT-nets into the OPS83 structures and the diagram of the system are shown in Table 1 and Figure 3.2 respectively. The BNF graph of the WPT-nets appeared in Appendix.

A. Introduction to OPS83

OPS83 is a rule-based language implemented

Table 1 The translation example of the WPT-nets into the OPS83 structures

A simple WPT-net	The BNF of simple WPT-net	Translate the simple WPT-net into the skeleton of the OPS83 structures
	pi ^ (- pj)→ pk	(P <transition_t_j> {(state^place_p_i{<marking_variable>≧ 1 } (state^place_p_j{< marking_variable>=0})} → (write (crlf) <transition_t_j> fired) (make (substr <state> 1 inf) ^vaild nil ^place_p_i(compute < marking_variable > -1) ^place_p_k(compute < marking_variable > +1)

in C language. Its earliest prototype was jus executing the function of the production system. Then it is used successfully to program includes the declaration part and the rulename part.

(a) Declaration part: It is used to declare the memory attributed that a program will use, and the function name adopted from other programming language that the user might employ.

(b) Rule part:

```
(P rulename
          (condition-1)
          (condition-2)
               .
               .
→
          (action-1)
          (action-2)
               .
               .
```

From the viewpoint of logic operation, the relationship between rules is : use "AND" to calculate in conditions of the same rule, but use "OR" to calculate in conditions of different rules.

B. Working memory of OPS83

All the dynamic materials of the OPS83 for program execution are gathered and managed in Working Memory(WM). Each unit in the working storage area is a Working Memory Element(WME). Each WME is composed of a time tag and a fact. The time tag is used to represent the time order of a fact joining in the WME. If the value is big or tends to be bigger and bigger, it means that it has been joined or modified recently.

C. Inference engine of OPS83

The operation of OPS83 inference engine is similar to production system. The OPS83 must find out at once, in rule memory, all rules which can satisfy its LHS(rulename part) after the user implements the initial material. The rule must be put into the Conflict Set(CS); so there is a slight difference between the order of the inference steps and that of the production system. The OPS83 execution inference steps are described as follows:

Step 1: Choose a proper rule from the CS.

Step 2: Execute the action part of this rule, and change the con-tents of the WME.

Step 3: Data match. Find out a rule matches LHS and put it in the CS.

Step 4: Coming across a "Halt" command. To stop inference if the CS is empty or actions executed to its

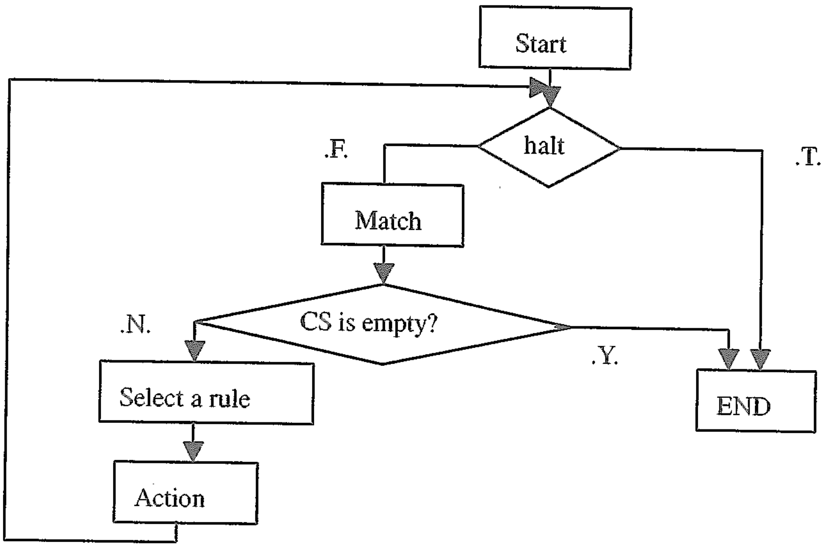


Figure. 3.2OPS83 execution cycle

limitation. Otherwise, go back to the step and keep on the inference. The OPS83 execution cycle is shown in Figure 3.1.

From the generalization narrate above, the characters between OPS83 and WPT-net are similar. Thus, the WPT-nets are translated into production rules, and use them to imitate, design, or other specially related issues.

The procedure for system design includes:

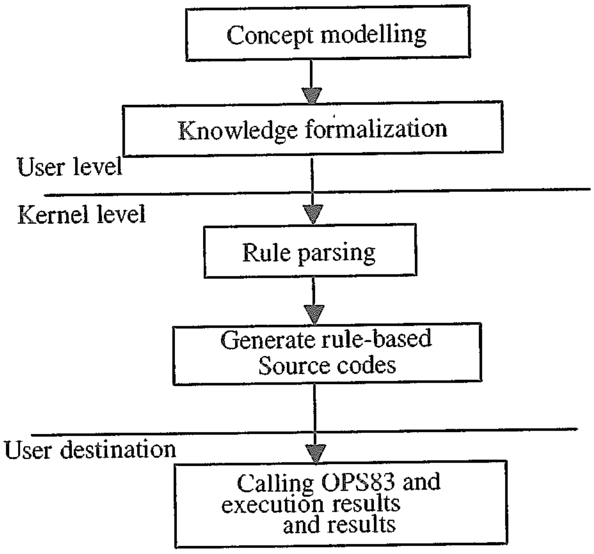


Figure 3.3 The diagram of the system design

(a) Knowledge formalization: To make the WPT-nets into production rules and use them as formal input specifications.

(b) Rule parsing: The rule parser checks out the correctness of the above input rules. Therefore, the system validation will be enhanced.

(c) Automatic generation source code According to the correct input rules, the system can automaticall produce rule -base program file by using a generator so that the execution of the production system will be convenient.

IV. Applications

This section illust ates the example of a workflow system described by WPT-nets to specify process control. The WPT-nets consists of subsystem A and subsystem B, while the translation of the WPT-nets into program structures concern resource conflict which may results from detecting deadlock and synchronization. The target language is OPS83, a rule-based program codes. An example of workflow system described by WPT -nets show in Figure 4.1.

A. User level

According to Figure 4.1, The WPT -nets are translated into production rules by using BNF graph(see Appendix), input the initial marking μ^0, and the deterministic timing of associating with transitions shown as below.

Concept modeling:

$P=\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, $T=\{t_1, t_2, t_3, t_4, t_5\}$

$I(t_1)=\{p_7\}$

$I(t_2)=\{p_1, p_6\}$

$I(t_3)=\{p_2\}$

$I(t_4)=\{p_3, p_5\}$

$I(t_5)=\{p_4, p_5\}$

$O(t_1)=\{p_1, p_2\}$

$O(t_2)=\{p_3, p_4\}$

$O(t_3)=\{p_5\}$

$O(t_4)=\{p_6\}$

$O(t_5)=\{p_7\}$

The input sequence of the production rules:

$t_1: p_7 \rightarrow p_1 \wedge p_2$

$t_2: p_1 \wedge p_6 \rightarrow p_3 \wedge p_4$

$t_3: p_2 \rightarrow p_5$

$t_4: p_3 \wedge p_5 \rightarrow p_6$

$t_5: p_4 \wedge p_5 \rightarrow p_7$

Input initial marki ng μ^0:

$(p_1, p_2, p_3, p_4, p_5, p_6, p_7)=(0, 0, 0, 0, 0, 1, 1)$

The deterministic timing of associating with

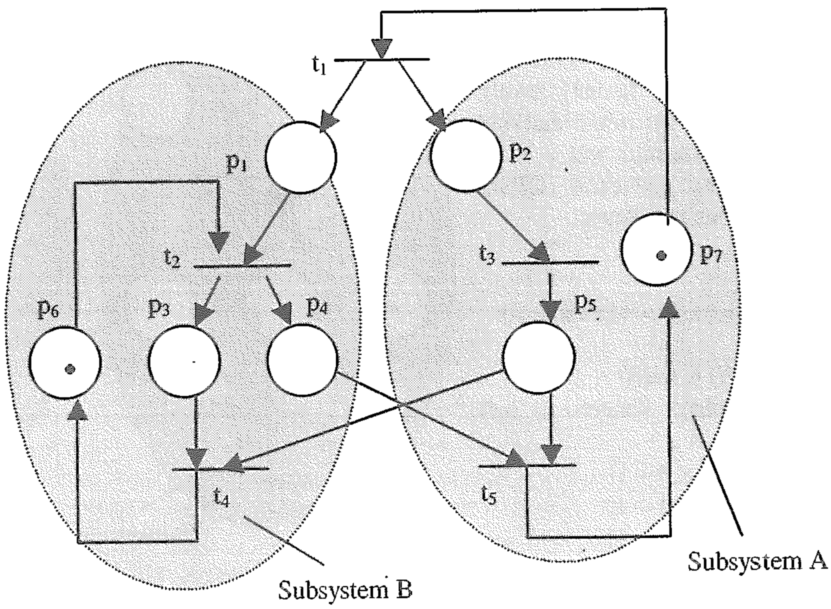


Figure 4.1 An example of workflow system described by WPT-net

transitions:

$\tau(t_1)=2, \tau(t_2)=2, \tau(t_3)=3, \tau(t_4)=2, \tau(t_5)=1$

B. Kernel level

The input rules can be translated into production rules. A OPS83 rule -based program skeleton that can be designed from by production rules shown as below(shadow parts are filled the places, transitions and the deterministic timing of associating with transitions that will be generated by rule parser.). Then the generation of the automatic rule -base program codes will be generated.

```
(P <transition_name>
{(state ^place_name_1 { < marking_variable_1
> ≧ constant_1 }) and
(state ^transition_name_1 {<transition_timing>
≧ constant_2})}
->
(write (crlf) <transition_name> fired)
(make (substr <state> 1 inf) ^vaild nil
^transition_name_1(compute
<transition_timing_1> -1)
^transition_name_2(compute
<transition_timing_2> -1)

.
.
^place_name_1(compute <marking_variable_1>
- constant_3)
^place_name_2(compute <marking_variable_2>
+ constant_4)
.
.
```

Now, an example is used to show the rule parser to fetch the place and transition nodes from the BNF of WPT-net in the Figure 4.1, and then fill them into the skeleton(shown kernel level). Finally, input the initial marking μ^0, the generator will automatically generate OPS83 source codes as follows:.

```
(P t1
{(state ^p6{ x_1 > 1}) <state>}
->
(write (crlf) t1 fired)
(make (substr <state> 1 inf)
^vaild nil
^p6(compute x_1 - 1)
^p1(compute y_1 + 1)
^p2(compute y_2 + 1)))
.
.
(P t2
{(state ^p1{ x_1 > 1}) <state>}
{(state ^p5{ x_1 > 1}) <state>}
->
(write (crlf) t2 fired)
(make (substr <state> 1 inf) ^vaild nil
^p3(compute y_1 +2)))
```

C. User destination level

According to the kernel level, the results are obtained and states of transitions are fired on many places. From upper case, the production system OPS83 can find out t_4 and t_5 can't fired(executed) simultaneously. Namely, t_4 and t_5 may occur deadlock (t_4 and t_5 are waiting each other release resource.). Therefore, the sequences of the transitions, places or transition firing time can be adjusted to avoid resource conflict. The resolutions will be discussed in detail as follows.

V. Discussion and strategies

In the kernel level, the generator of the OPS83 source codes that can detect t_4 and t_5 will occu reso-urce conflict and synchronization problems. Several strategies are presented to handle these issues and using the rapid prototyping method to redesign the workflow systems.

(a) Problem 1: Resource conflict

Strategy (i): Increase resources. The resource distribution is adjusted in the systems. For instance, one(or several) token is added into p_2, p_5 and p_7 respectively, but increasing the system cost. The workflow system diagr am shown as Figure 4.2.

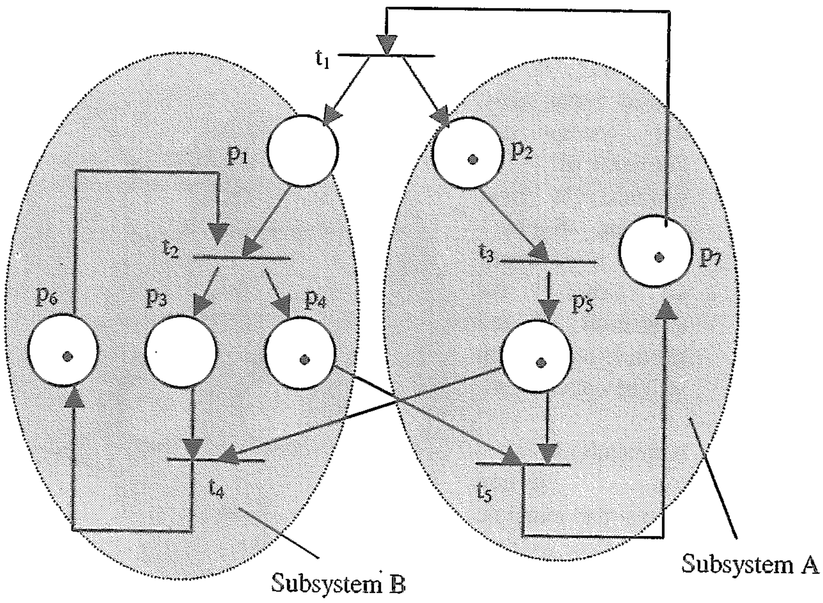


Figure 4.2 Add one token into p_2, p_5 and p_7 for solving resourceconflict between t_4 and t_5

Strategy (ii): Inhibitor arc extension. The Mutual Exclusive Relationship(p_5 between t_4 and t_5) is took place into inhibitor arc mechanism for zero testing p_5 that can solve the resource conflict problem and don't need any ex tra resource addiction however, increasing the programming overhead. The workflow syste diagram shown in Figure 4.3.

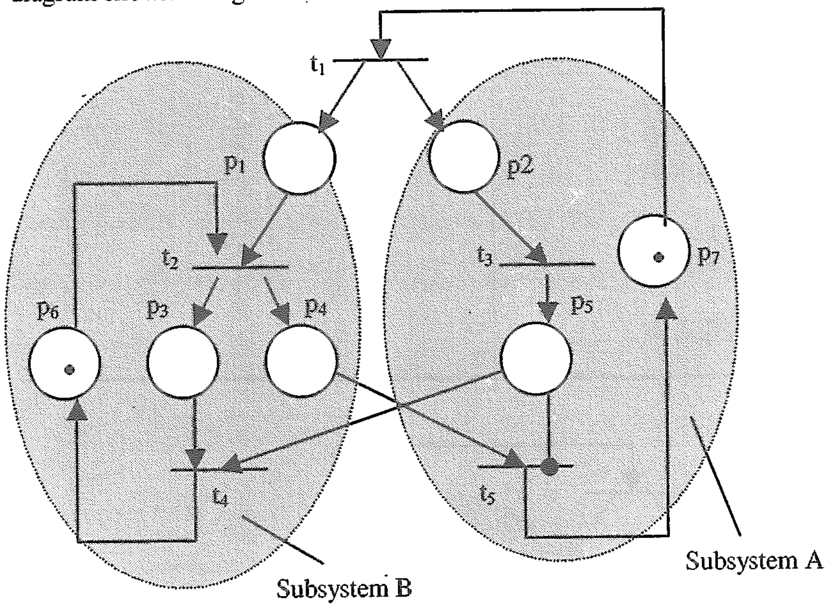


Figure 4.3 The workflow system diagram of inhibitor arc taking place the Mutual Exclusive Relationship

(a) Problem 2: Serialization issues

Strategy (i): Adjust timing. The transition firing sequence and using deterministic timing of associating with transitions is specified to reduce the side effect of Mutual Exclusive Relationship in the WPT-nets. The serialization of the syste processing is to prevent infinite loop(so called deadlock), shared resource conflict, or even the system crash that no one knows. The firing instances of deterministic timing of associating with transitions shown in Table 2.

Strategy (ii): Reorganize the syste specifications. The main character of the rapid prototyping is easily to simulate the syste behaviors, turning the processes performed sequence and share resource. In following example, we copy a new place p_5' from $_5$ to take place Mutual Exclusive Relationship between t_4 and t_5 that can reduce side effec because of resource sharing conflict. The reorganized system diagram shown in Figur 4.4.

VI. Conclusion

In this paper, the WPT-nets have been presented for the purpose of rule -based prototyping activities in the field of workflow systems. The WPT-net model can support the process approach to requirement s pecifications and is suitable for carrying out the syste simulations. The authors have shown how to translate into rule-based program structures by using the WPT-nets.

In order to estimate the accurate execution times,[2] further researches of WPT -nets are necessary. Another difficulty associated with the proposed approach may be the inaccuracy in estimating the execution times of the workflow processes in the production system. In this paper, the execution sequences of production rules used the time tag and firing strategies of the rule-based. However, in real time systems, execution time of processes must be estimated real execution time in the production rules.

An effective methodology[3] depends on the user-friendly of the supporting environment, so current work is devoted to the implementation of an integrated software environment which can manage in a common database including whole SDLC(Software Development Life Cycle).

Appendix

The BNF graph of the WPT-nets

<I.R.> ::= <T.N.> : <I.P.> <O.S.> <O.P.>
<T.N.> ::= <N><n>
<N> ::= "A"| …| "Z" | "a" | …|"z"
<n> ::= "1"| …| "9" | "A" | …|"Z"
<I.P.> ::= P <n1> [<A.S.> <I.P.>]
<n1> ::= 3 | 4 | …| 998
<A.S.> ::= "^"
<O.S.> ::="→"
<O.P.> ::= P<n2> [<A.S.> <O.P.>]
<n2> ::= 3 | 4 | …| 998

<I.R.> is Input Rule
<T.N.> is Transition Name
<I.P.> is Input Place
<A.S.> is "And"operation Symbol
<O.S.> is Output Symbol

<O.P.> is Output Place

Table 2 The firing instances of deterministic timing of associating with transitions

	Place	Transition fired time
	(p_1 p_2 p_3 p_4 p_5 p_6 p_7)	t_1 t_3 t_2 t_4 t_5
Time=0	(0, 0, 0, 0, 0, 1, 1)	- - - - -
waiting		2 - - - -
Time=1	(0, 0, 0, 0, 0, 1, 1)	1 - - - -
Time=2	(1, 1, 0, 0, 0, 1, 0)	- - - - -
waiting		- - - - -
waiting		- 2 3 - -
Time=3	(1, 1, 0, 0, 0, 1, 0)	- 1 2 - -
Time=4	(1, 0, 0, 0, 1, 1, 0)	- - 1 - -
Time=5	(0, 0, 1, 1, 1, 0, 0)	- - - - -
waiting		- - - 2 -
Time=6	(0, 0, 1, 1, 1, 0, 0)	- - - 1 -
Time=7	(0, 0, 0, 1, 1, 2, 0)	- - - - -
waiting		- - - - 1
Time=8	(0, 0, 0, 0, 0, 2, 2)	- - - - -

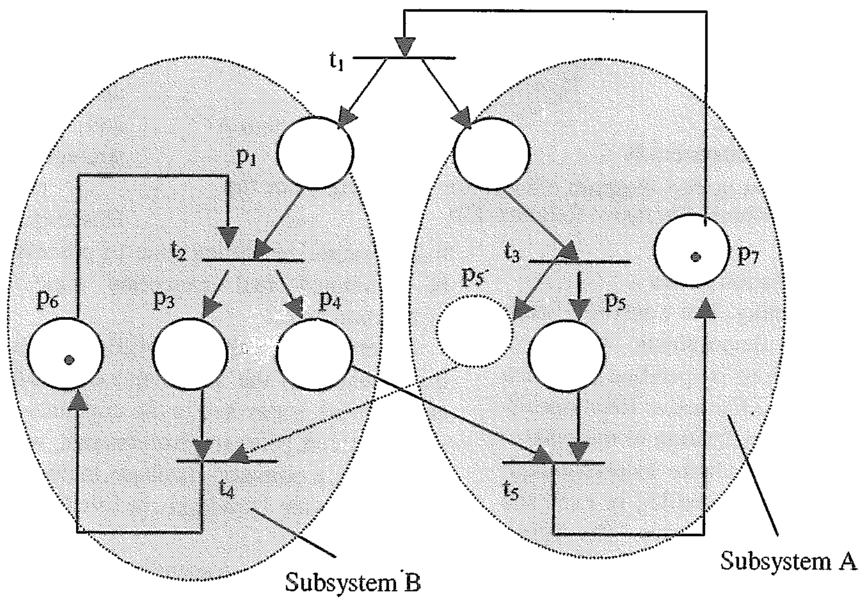


Figure 4.4 The new system diagram of copying a new place p_5' from p_5

REFERENCES

[1] Boujarwah Abdulaziz and Ai-seif Nadia and Saleh Kassem, "Modelling the semantics of multitasking facilities in concurrent C using Petri Nets," Information and Software Technology, Vol. 38, Iss. 1, pp. 3 -9, Jan. 1996.

[2] Perkusich Angelo and De Figueiredo Jorge, "G-nets: a Petri Net based approach for logical and timing analysis of complex software system," Journal of System and Software, Vol. 39 Iss. 1, pp. 39-59, Oct. 1997.

[3] Rakeh Agrawal, Michael J. Carey and LawrenceW. Mcvoy, "The performance of alternative strategies for dealing with deadlocks in database management systems, " IEEE Trans., Software Eng. Vol. Se-13, no. 12, pp. 1384-1363, Dec.,1987.

[4] Lee Brownston, Robert Farrell, Elaine Kant, and Nanvy Martin, "Programming expert systems in OPS5, A introduction to rule based programming, " Addison-Wesley publishing company, inc., 1985.

[5] Giorgio Bruno and Giuseppe Marc hetto, "Process-Translatable Petri Nets for the Rapid Prototyping of Process Control Systems," IEEE Trans. On Software Eng. Vol. SE-12, no. 2, 346-357, Feb. 1986.

[6] Jenn-Nan Chen, "Verification and

translation of distributed computing system software design," Phd. Dissertation, June 1987, Northwestern University.

[7] Jenn-Nan Chen and Pi-Te Chen,"Diagnosis system for automatic detection of deadlock in asynchronous concurrent distributed computing systems : using Timed Petri Net with stacks," Proc. Internati nal IEEE Conf. on COMSAC, pp.658-664,1990.

[8] J. Duggan, and J. Browne, "ESPNET : expert-system-based simulator of Petri nets," IEE Proc., Vol. 135, Pt.D, no. 4, July 1988.

[9] Peterson J. L., Petri Net theory and the modelling of systems, Prentice-Hall, Englewood Cliffs, Nj., 1981.

[10] Jan Magott, "New NP-complete problems in performance evaluation of concurrent system using Petri Nets," IEEE Trans Software Eng. Vol. Se-13, no.5, pp. 578-581, May 1987.

[11] Krishna m. Kavi, Billy P. Buckles and U. Narayan Bhat,"Isomorphisms between Petri Nets and dataflow graphs, " IEEE Trans Software Eng. Vol. Se-13, no. 10, pp. 1127-1134, Oct. 1987.

[12] C. Y. Wong, t. s. Dillon and K. E. Forward, "Concurrent, real time systems : a systematic approach using timed Petri Nets," Computer System Science and Eng. Vol. 2, no. 3, pp. 1 -124, July 1987.

[13] Yao Weili and He Xudong, "Mapping Petri Nets to concurrent programs in CC++," Information and Software Technology, Vol. 39, Iss. 7, pp. 485-495, July, 1997.