Acta Technica *62*, No. 3B/2017, 53–62 © 2017 Institute of Thermomechanics CAS, v.v.i.

# Extending concurrency within component net on the basis of EEOOPN[1]

Na Zhao[2], Tong Li[2], Yong Yu[2,4], Jian Wang[3], Zhongwen Xie[2], Jinzhuo Liu[2]

**Abstract.** One efficient way to speed up component evolution process is to evolve it in a concurrent way. By employing EEOOPN, an Extended Object-Oriented Petri Net software component evolution modeling tool presented in authors' previous effort, a methodology of extending concurrency globally is proposed. By means of dependence analysis between partition blocks and extending the concurrency on the basis of concurrency factors searching and capturing, the overall efficiency of component evolvement is achieved through this approach.

**Key words.** Component net, EEOOPN, dependence analysis, transition, partition block, synchronization relation..

## 1. Introduction

Petri net can be applied formally or informally in most system to described it graphically and to represent parallel or concurrent activities. The main advantage of Petri net lies in the fact that the system concurrency are captured both on conceptual and mathematical basis [1]. Since the appealing graphical representation Petri net has, a lot of Petri net modeling tools were developed. EEOOPN [2], an Extended Object-Oriented Petri Net, is such a modeling tool proposed by the authors of this

[1]This work has been supported by the Data Driven Software Engineering innovation team of Yunnan province, by the Science Foundation of Yunnan Province, China under Grant No. 2016FB111, 2016FB102, 2015FA014, by the Science Foundation of Key Laboratory in Software Engineering of Yunnan Province under Grant No. 2017SE205, 2017SE201, 2016SE202, by the education Foundation of School of Software under Grant No. 2014EI03.

[2]School of Software, Key Laboratory in Software Engineering of Yunnan Province, Yunnan University, Kunming, China

[3]College of Information Engineering and Automation, Kunming University of Science and Technology, China

[4]Corresponding author

paper. EEOOPN can be exploited for component-based software system modeling, simulation and analysis of the structural and behavioral properties. EEOOPN also provide users a natural way to capture many of the basic notions and issues of a concurrent system.

The focus of this paper is the time performance of component nets. To be specific, the concurrency in component evolution processes. The objective is to extending concurrency between component nets. This objective is achieved by reconstruction of component net on the basis of dependence analysis between partitions of transitions and extending the concurrency form the local to the global.

## 2. Relative works

Petri net is an essential modeling language used to describe a system graphically and mathematically. It is used to model control flow in a system and is capable of modeling concurrency and synchronization. By allowing the definition of active concurrent objects, CoopnBuilder, a Concurrent Object Oriented Petri Nets tool, is designed to support concurrent software development based on synchronized Algebraic Petri net [3].

Literature [4] presented a hierarchical colored Petri net model of TO, timestamp ordering method, to preserve concurrency between concurrent components. Alekseyev A et al. [5] propose an optimised algorithm for computing the parallel composition including the modeling and evaluation of interacting parallel components, which facilitates the subsequent concurrency evolvement. Literature [6] introduced concepts that make possible working with concurrency semantics via (safe) Petri net and occurrence nets based on the exploitation of the partial order semantics and essential dependencies analysis.

Tong Li proposed an object-oriented concurrent evolutionary software development model [7]. A Petri Net-based concurrency processes model was presented in literature [8] by the excavating of concurrent factors through activity dependences analysis. In the meanwhile, concurrency is extended from local to global.

## 3. Component net properties

**Definition 1:**

In a component system of EEOOPN $CN_{\mathrm{E}} = (P_{\mathrm{E}}, T_{\mathrm{E}}, F_{\mathrm{E}}, S_{\mathrm{E}}, A_{\mathrm{PE}}, A_{\mathrm{TE}}, A_{\mathrm{FE}}, M_0)$, $\forall p \in P_{\mathrm{E}}, \forall t \in T_{\mathrm{E}}$, $p$ and $t$ is type matched if $A_{\mathrm{FE}}(pt) \in L_{\mathrm{m}}(A_{\mathrm{PE}}(p))$ is true when $(pt) \in F_{\mathrm{E}}$ or $(tp) \in F_{\mathrm{E}}$, if $A_{\mathrm{FE}}(tp) \in L_{\mathrm{m}}(A_{\mathrm{PE}}(p))$ is true when $(tp) \in F_{\mathrm{E}}$.

**Definition 2:**

Let $C_i{\cdot}oip_{\mathrm{E}}$ be the interface of a component net and let $LN_{\mathrm{E}}{\cdot}t_{\mathrm{in}}$, $LN_{\mathrm{E}}{\cdot}t_{\mathrm{out}} \in LN_{\mathrm{E}}$ be the entrance and exit transition of a connector $LN_{\mathrm{E}}$, if $A_{\mathrm{FE}}(C_i \cdot oip_{\mathrm{E}}, LN_{\mathrm{E}} \cdot t_{\mathrm{in}}) \in L_{\mathrm{m}}(A_{\mathrm{PE}}(C_i \cdot oip_{\mathrm{E}}))$ and $A_{\mathrm{FE}}(LN_{\mathrm{E}} \cdot t_{\mathrm{out}}, C_i \cdot oip_{\mathrm{E}}) \in L_{\mathrm{m}}(A_{\mathrm{PE}}(C_i \cdot oip_{\mathrm{E}}))$, then component interface $C_i \cdot oip_{\mathrm{E}}$ and connector $LN_{\mathrm{E}}$ is called type matched.

**Definition 3:**

A well-structured $CN_{\mathrm{E}} = (P_{\mathrm{E}}, T_{\mathrm{E}}, F_{\mathrm{E}}, S_{\mathrm{E}}, A_{\mathrm{PE}}, A_{\mathrm{TE}}, A_{\mathrm{FE}})$ satisfies:

1. $|CN_{\mathrm{E}} \cdot oip| = 1$;
2. For $\forall p \in P_{\mathrm{E}}, \forall t \in T_{\mathrm{E}}$, $p$ and $t$ is type matched if $(pt) \in F_{\mathrm{E}}$ or $(tp) \in F_{\mathrm{E}}$;
3. For $\forall t \in T_{\mathrm{E}}$ and $\forall p_i \in {}^{\bullet}t(i = 1, 2, \cdots, n)$, then $\sum_{i=1}^{n} A_{\mathrm{FE}}(p_i, t) = A_{\mathrm{TE}}(t)$.

## 4. Using dependence matrix of component net to express transition dependence

**Definition 4:**

Let $\sum = (P_{\mathrm{E}}, T_{\mathrm{E}}, F_{\mathrm{E}}, S_{\mathrm{E}}\, A_{\mathrm{PE}}, A_{\mathrm{TE}}, A_{\mathrm{FE}})$ be a component net of EEOOPN, the structure of $\sum$ can be represent by a $n \times m$ matrix

$$A_{\mathrm{E}ij} = [a_{ij}]_{n \times m}\,, \tag{1}$$

where $a_{ij}$ is a $k$-dimensional integral vector $a_{ij} = a_{ij}^{+} - a_{ij}^{-}, i \in \{1, 2, \cdots, n\}, j \in \{1, 2, \cdots, m\}$,

$$a_{ij}^{+} = \begin{cases} A_{\mathrm{FE}} t_i,\, s_j,\, if\ t_i,\, s_j \in F_{\mathrm{E}}, \\ 0, \qquad\qquad \text{other cases} \end{cases} i \in \{1, 2, \cdots, n\} j \in \{1, 2, \cdots, m\}\,, \tag{2}$$

$$a_{ij}^{-} = \begin{cases} A_{\mathrm{FE}} s_j,\, t_i,\, if\ s_j, t_i \in F_{\mathrm{E}}, \\ 0, \qquad\qquad \text{other cases} \end{cases} i \in \{1, 2, \cdots, n\} j \in \{1, 2, \cdots, m\}\,. \tag{3}$$

$A_{\mathrm{E}}$ is called the associated matrix of $\sum$. The matrix of $A_{\mathrm{E}}^{+} = [a_{ij}^{+}]_{n \times m}$ and $A_{\mathrm{E}}^{-} = [a_{ij}^{-}]_{n \times m}$ is called input and output matrix of $\sum$ respectively.

Respectively, $A_{\mathrm{E}i*}$, $A_{\mathrm{E}i*}^{+}$ and $A_{\mathrm{E}i*}^{-}$ are used to represent row vector of $i$-th row of matrix $A_{\mathrm{E}}$, $A_{\mathrm{E}}^{+}$, $A_{\mathrm{E}}^{-}$ while using $A_{\mathrm{E}*j}$, $A_{\mathrm{E}*j}^{+}$ and $A_{\mathrm{E}*j}^{-}$ to present $j$-th column vector of matrix $A_{\mathrm{E}}$, $A_{\mathrm{E}}^{+}$, $A_{\mathrm{E}}^{-}$.

The relation between transitions can be represented by the dependence matrix. Each row of dependence matrix corresponds to the dependence between a transition $t$ and its corresponding place set. If a row of a dependence matrix, which is a $k$-dimensional vector $a_{ij}$, contains more than one negative elements, then the firing of the transition needs all the places $p$ satisfying the condition of $A_{\mathrm{PE}}(p) \geq A_{\mathrm{FE}}(p, t)$. The vectors here represented the synchronization relation between the $t$ and the place set. If a row contains more than one positive element, then the corresponding places are mutually independent. This represents the concurrency relation between the $t$ and the place set.

Each column of dependence matrix corresponds to the dependence between a place $p$ and its corresponding transition set. If a column of a dependence matrix, which is a $k$-dimensional vector $a_{ij}$, contains more than one negative elements, then $A_{\mathrm{PE}}(p) < \sum_{t \in p^{\bullet}} A_{\mathrm{FE}}(p, t)$, which represents the competitive relation between the transitions and the resources possessed by the place. $\sum_{t \in {}^{\bullet}p} A_{\mathrm{FE}}(t, p) > A_{\mathrm{PE}}(p)$ represent the output dependence because it means the place will get a *token* if an arbitrary $t$

in the transition set is fired.

## 5. Analyzing dependences between component nets

As shown in Fig. 1, $CN_{\rm A} = (P_{\rm EA}, T_{\rm EA}, F_{\rm EA}, S_{\rm EA}, A_{\rm PEA}, A_{\rm TEA}, A_{\rm FEA})$ and $CN_{\rm B} = (P_{\rm EB}, T_{\rm EB}, F_{\rm EB}, S_{\rm EB}, A_{\rm PEB}, A_{\rm TEB}, A_{\rm FEB})$ are two synchronistic component nets, each of the two component net is refined as many concurrent transitions (e.g. $a_1, a_2, \cdots, a_n \in T_{\rm EA}$ and $b_1, b_2, \cdots, b_m \in T_{\rm EB}$). However, this concurrency in each of the component net is local. Thus, the global synchronization relation between transition $a_1, a_2, \cdots, a_n \in T_{\rm EA}$ and transition $b_1, b_2, \cdots, b_m \in T_{\rm EB}$ needs to be degraded to local synchronization. Namely, extends the local concurrency to global concurrency. During the processes of concurrency extending, synchronization relation must be remained.

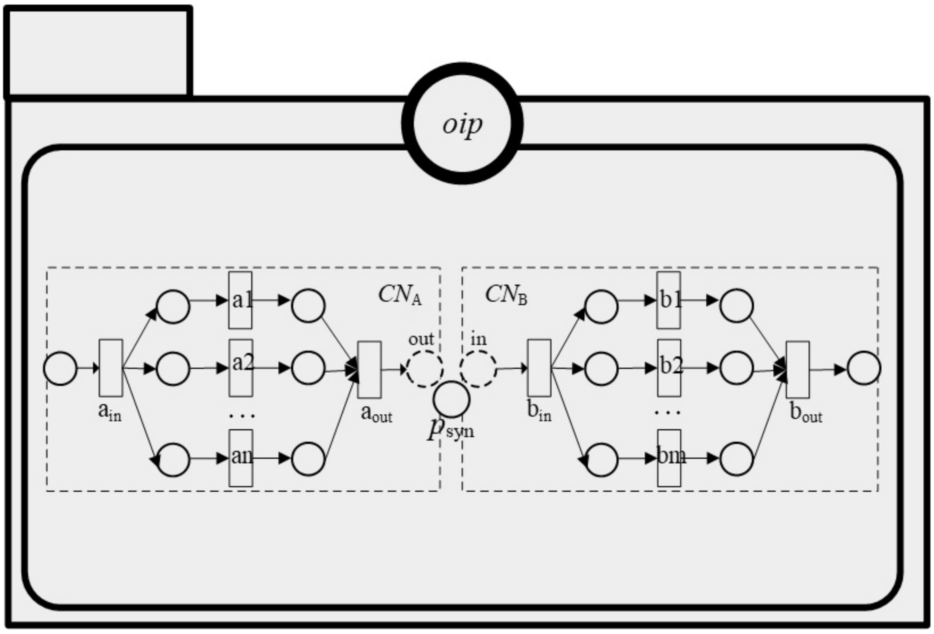

Fig. 1. Two component nets with synchronization relation

**Definition 5:**

Let $A_{\rm E} = \{a_1, a_2, \ldots, a_n\}$ be a transition set in component net of EEOOPN. Relation $R_{\rm E}$ is called a synchronization relation on $A_{\rm E}$ iff.

For $x, y \in A_{\rm E}$:

1. If $x$ and $y$ must be executed synchronously, $(x, y) \in R_{\rm E} \wedge (y, x) \in R_{\rm E}$;
2. $(x, x) \in R_{\rm E}$;
3. If $(x, y) \in R_{\rm E} \wedge (y, z) \in R_{\rm E}$, $(x, z) \in R_{\rm E}$.

**Theorem 1:**

Synchronization relation $R_{\mathrm{E}}$ is an equivalence relation on $A_{\mathrm{E}}$. The conclusion is obvious.

Given the fact that $R_{\mathrm{E}}$ is an equivalence relation on $A_{\mathrm{E}}$, one can construct the equivalence classes of $A_{\mathrm{E}}$ and obtain a partition $\{Ab_{\mathrm{E}1}, Ab_{\mathrm{E}2}, \ldots, Ab_{\mathrm{E}k}\}$, where $Ab_{\mathrm{E}i}$ $(i = 1, 2, \ldots, k)$ denotes a partition block, each of which corresponds to a component net of EEOOPN.

Let $Ab_{\mathrm{E}i}$ and $Bb_{\mathrm{E}j}$ be partition blocks. The concurrency relationship between partition blocks $Ab_{\mathrm{E}i}$ and $Bb_{\mathrm{E}j}$ can be determined by examining the following rules:

1. $Ab_{\mathrm{E}i}$ and $Bb_{\mathrm{E}j}$ can only be executed sequentially when $Bb_{\mathrm{E}j}$ depends on $Ab_{\mathrm{E}i}$;
2. $Ab_{\mathrm{E}i}$ and $Bb_{\mathrm{E}j}$ can be executed concurrently when $Bb_{\mathrm{Ej}}$ is independent of $Ab_{\mathrm{E}i}$.

**Algorithm 1:**

Dependence Analysis between two Partition Blocks

Input: $CN_{\mathrm{A}} = (P_{\mathrm{EA}}, T_{\mathrm{EA}}, F_{\mathrm{EA}}, S_{\mathrm{EA}}, A_{\mathrm{PEA}}, A_{\mathrm{TEA}}, A_{\mathrm{FEA}})$, $CN_{\mathrm{B}} = (P_{\mathrm{EB}}, T_{\mathrm{EB}}, F_{\mathrm{EB}}, S_{\mathrm{EB}}, A_{\mathrm{PEB}}, A_{\mathrm{TEB}}, A_{\mathrm{FEB}})$, transition set $A_{\mathrm{E}} = \{a_1, a_2, \ldots, a_n\}$, the synchronization relation $R_{\mathrm{EA}}$ on $A_{\mathrm{E}}$, transition set $B_{\mathrm{E}} = \{b_1, b_2, \ldots, b_m\}$, the synchronization relation $R_{\mathrm{EB}}$ on $B_{\mathrm{E}}$.

Output: transition set $Ab_{\mathrm{E}}$, $Bb_{\mathrm{E}}$, two-dimensional array $D_{\mathrm{E}}$ that shows the dependence between transition sets.

```
BEGIN
  Construct quotient set $A_E/R_{EA}$, obtain transition set $Ab_E = \{Ab_{E1}, Ab_{E2}, \cdots, Ab_{Es}\}$;
  Construct quotient set $A_E/R_{EB}$, obtain transition set $Bb_E = \{Bb_{E1}, Bb_{E2}, \cdots, Bb_{Et}\}$;
  $s := |A_E/R_{EA}|$;
  $t := |B_E/R_{EB}|$;
  FOR $i := 1$ TO $s$ DO
    FOR $j := 1$ TO $t$ DO
      BEGIN
        $D_E\,[i, j] := false$;
        $na_E := |Ab_{Ei}|$;
        $nb_E := |Bb_{Ej}|$;
        FOR $k := 1$ TO $na_E$ DO
          FOR $l := 1$ TO $nb_E$ DO
            Call dependence analysis algorithm, obtain dependence between transitions;
            IF $a_{ik}, b_{jl}$ is dependent THEN $D_E\,[i, j] := true$;
      END;
END.
```

## 6. Extending of concurrency between component nets

After finishing the dependence analysis of partition blocks of $Ab_{\mathrm{E}i}$ and $Bb_{\mathrm{E}j}$, one can start the process of reconstructing EOOPN component net to extend the concurrency.

**Algorithm 2:**

Extending of concurrency between component nets of EEOOPN.

Input: $CN_{\mathrm{E}} = (P_{\mathrm{E}}, T_{\mathrm{E}}, F_{\mathrm{E}}, S_{\mathrm{E}}, A_{\mathrm{PE}}, A_{\mathrm{TE}}, A_{\mathrm{FE}})$ shown in Fig. 1, transition set $A_{\mathrm{E}} = \{a_1, a_2, \ldots, a_n\}$, the synchronization relation $R_{\mathrm{EA}}$ on $A_{\mathrm{E}}$, transition set $B_{\mathrm{E}} = \{b_1, b_2, \ldots, b_m\}$, the synchronization relation $R_{\mathrm{EB}}$ on $B_{\mathrm{E}}$, $CN_{\mathrm{E}} = CN_{\mathrm{A}} \bigcup CN_{\mathrm{B}}$, $CN_{\mathrm{A}} \bigcap CN_{\mathrm{B}} = p_{syn}$.

Output: Component net $CN'_{\mathrm{E}} = (P'_{\mathrm{E}}, T'_{\mathrm{E}}, F'_{\mathrm{E}}, S'_{\mathrm{E}}, A'_{\mathrm{PE}}, A'_{\mathrm{TE}}, A'_{\mathrm{FE}})$ with extended concurrency.

```
BEGIN
  Call dependence analysis algorithm 1 to obtain dependence array $D_{\mathrm{E}}$, partitions $A_{\mathrm{E}}/R_{\mathrm{AE}}$
  and $B_{\mathrm{E}}/R_{\mathrm{BE}}$.
  $s := |Ab_{\mathrm{E}}|$;
  $t := |Bb_{\mathrm{E}}|$;
  $P'_{\mathrm{E}} := P_{\mathrm{E}} \backslash p_{\mathrm{syn}}$;
  $T'_{\mathrm{E}} := T_{\mathrm{E}} \backslash \{{}^{\bullet}p_{\mathrm{syn}}, p_{\mathrm{syn}}{}^{\bullet}\}$;
  $F'_{\mathrm{E}} := F_{\mathrm{E}} \backslash \{\bigcup_{p \in {}^{\bullet\bullet}p_{\mathrm{syn}}} (p, {}^{\bullet}p_{\mathrm{syn}}), \bigcup_{p \in p_{\mathrm{syn}}{}^{\bullet\bullet}} (p^{\bullet}_{\mathrm{syn}}, p)\}$;
  $S'_{\mathrm{E}} := S_{\mathrm{E}}$;
  $A'_{\mathrm{PE}} := A_{\mathrm{PE}}$;
  $A'_{\mathrm{TE}} := A_{\mathrm{TE}}$;
  $A'_{\mathrm{FE}} := A_{\mathrm{FE}}$;
  $A'_{\mathrm{FE}} := A_{\mathrm{FE}} \backslash \{\bigcup_{p \in {}^{\bullet\bullet}p_{\mathrm{syn}}} A_{\mathrm{FE}}(p, {}^{\bullet}p_{\mathrm{syn}}), \bigcup_{p \in p_{\mathrm{syn}}{}^{\bullet\bullet}} A_{\mathrm{FE}}(p^{\bullet}_{\mathrm{syn}}, p)\}$;
  FOR $i := 1$ TO $s$ DO
  BEGIN
    $na_{\mathrm{E}} := |Ab_{\mathrm{E}i}|$;
    IF $D_{\mathrm{E}}[i, 1, \cdots, t]$ = false THEN
      BEGIN
        FOR $k := 1$ TO $na_{\mathrm{E}}$ DO
          $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Ab_{\mathrm{E}i}} (t, b_{\mathrm{out}}) \backslash \bigcup_{t \in Ab_{\mathrm{E}i}} (t, {}^{\bullet}p_{\mathrm{syn}})$ ;
          $\forall f \in \bigcup_{t \in Ab_{\mathrm{E}i}} (t, b_{\mathrm{out}})$, $A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(t, {}^{\bullet}p_{\mathrm{syn}})$;
      END;
    ELSE
      FOR $j := 1$ TO $t$ DO
        IF $D_{\mathrm{E}}[i, j]$ THEN
          BEGIN
            $P'_{\mathrm{E}} := P'_{\mathrm{E}} \bigcup c_{ij}$;
            $T'_{\mathrm{E}} := T_{\mathrm{E}} \bigcup \{b'_j, a''_i\}$;
            $nb_{\mathrm{E}} := |Bb_{\mathrm{E}j}|$;
            FOR $l := 1$ TO $nb_{\mathrm{E}}$ DO
              $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Bb_{\mathrm{E}i}} (b'_j, {}^{\bullet}t)$;
              $\forall f \in \bigcup_{t \in Bb_{\mathrm{E}i}} (b'_j, {}^{\bullet}t), A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(p_{\mathrm{syn}}{}^{\bullet}, {}^{\bullet}t)$;
            FOR $k := 1$ TO $na_{\mathrm{E}}$ DO
              $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Ab_{\mathrm{E}i}} (t^{\bullet}, a''_i); \forall f \in \bigcup_{t \in Ab_{\mathrm{E}i}} (t^{\bullet}, a''_i), A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(t^{\bullet}, {}^{\bullet}p_{\mathrm{syn}})$;
              $A'_{\mathrm{TE}}(a''_i) := \sum_{p \in {}^{\bullet}a''_i} (p, a''_i)$;
          END;
    END;
    FOR $j := 1$ TO $t$ DO
      IF $D_{\mathrm{E}}[1, \cdots, s, j] = false$ THEN
        BEGIN
          $nb_{\mathrm{E}} := |Bb_{\mathrm{E}j}|$;        FOR $l := 1$ TO $nb_{\mathrm{E}}$ DO
            $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (a_{\mathrm{in}}, p) \backslash \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (p_{\mathrm{syn}}{}^{\bullet}, p)$;
          $\forall f \in \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (a_{\mathrm{in}}, p), A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(p_{\mathrm{syn}}{}^{\bullet}, p)$;
        END;
    $A'_{\mathrm{TE}}(b_{\mathrm{out}}) := \sum_{p \in {}^{\bullet}b_{\mathrm{out}}} (p, b_{\mathrm{out}})$;
END.
```

In algorithm 2, if there is no dependence between a partition block in $A_{\mathrm{E}}$ and any partition block in $B_{\mathrm{E}}$, then those arcs pointing to $p_{\mathrm{syn}}$ will be changed to point to $b_{\mathrm{out}}$. If there is no dependence between a partition block in $B_{\mathrm{E}}$ and any partition block in $A_{\mathrm{E}}$, then those arcs pointing to the block from $p_{\mathrm{syn}}$ will be changed to point to the block from $a_{\mathrm{in}}$.

Algorithm is shown in Fig. 2.

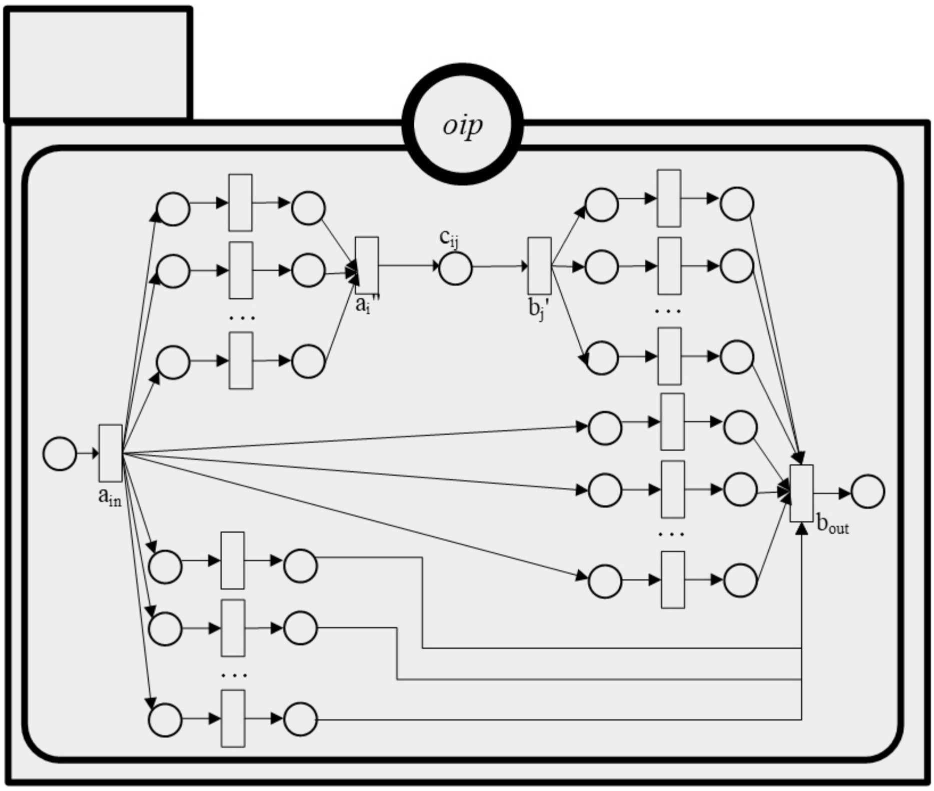

Fig. 2. Extending concurrency on Fig. 1

**Theorem 2:**

Let $CN_{\mathrm{A}}$ and $CN_{\mathrm{B}}$ be two well-structured component nets in well-structured component net $CN_{\mathrm{E}} = (P_{\mathrm{E}}, T_{\mathrm{E}}, F_{\mathrm{E}}, S_{\mathrm{E}}, A_{\mathrm{PE}}, A_{\mathrm{TE}}, A_{\mathrm{FE}})$. The component net $CN'_{\mathrm{E}} = (P'_{\mathrm{E}}, T'_{\mathrm{E}}, F'_{\mathrm{E}}, S'_{\mathrm{E}}, A'_{\mathrm{PE}}, A'_{\mathrm{TE}}, A'_{\mathrm{FE}})$ whose concurrency has been extended by algorithm 2, is also well-structured.

**Proof:**

The algorithm 2 exert no changes on the $oip_{\mathrm{E}}$ of $CN'_{\mathrm{E}}$. Namely the interface of $CN'_{\mathrm{E}}$ remain unchanged, so $CN'_{\mathrm{E}}$ meets the well-structured property (1).

According to the operations described in algorithm 2: $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Ab_{\mathrm{E}i}} (t, b_{\mathrm{out}}) \setminus \bigcup_{t \in Ab_{\mathrm{E}i}} (t, {}^{\bullet}p_{\mathrm{syn}})$, $\forall f \in \bigcup_{t \in Ab_{\mathrm{E}i}} (t, b_{\mathrm{out}})$, $A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(t, {}^{\bullet}p_{\mathrm{syn}})$, $T'_{\mathrm{E}} := T_{\mathrm{E}} \bigcup$

$\{b'_j,\, a''_i\}$, $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Bb_{\mathrm{E}i}} (b'_j,\, {}^{\bullet}t)$, $\forall f \in \bigcup_{t \in Bb_{\mathrm{E}i}} (b'_j,\, {}^{\bullet}t)$, $A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(p_{\mathrm{syn}}{}^{\bullet},\, {}^{\bullet}t)$, $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{t \in Ab_{\mathrm{E}i}} (t^{\bullet},\, a''_i)$, $\forall f \in \bigcup_{t \in Ab_{\mathrm{E}i}} (t^{\bullet},\, a''_i)$, $A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}(t^{\bullet},\, {}^{\bullet}p_{\mathrm{syn}})$, $F'_{\mathrm{E}} := F'_{\mathrm{E}} \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (a_{\mathrm{in}},\, p) \backslash \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (p_{\mathrm{syn}}{}^{\bullet},\, p)$, $\forall f \in \bigcup_{p \in {}^{\bullet}Bb_{\mathrm{E}i}} (a_{\mathrm{in}},\, p)$, $A'_{\mathrm{FE}}(f) := A_{\mathrm{FE}}$ $(p_{\mathrm{syn}}{}^{\bullet},\, p)$, the modification of arcs does not change the type-matched property between $p'$ and $t'$. Therefore, for $\forall t \in T'_{\mathrm{E}}$ and $\forall p \in P'_{\mathrm{E}}$, if $(pt) \in F'_{\mathrm{E}}$ or $(tp) \in F'_{\mathrm{E}}$, then the $p'$ and $t'$ in $CN'_{\mathrm{E}}$ is type-matched. This proves $CN'_{\mathrm{E}}$ meets the second property of well-structured component.

Given component net $CN_{\mathrm{E}}$ is well-structured, according to the third property of well-structured component, for $\forall t \in T_{\mathrm{E}}$ and $\forall p_i \in {}^{\bullet}t (i = 1, 2, \cdots, n)$, one will have $\sum\limits_{i=1}^{n} A_{\mathrm{FE}}(p_i,\, t) = A_{\mathrm{TE}}(t)$, and also according to the operations of $A'_{\mathrm{TE}}(a''_i) := \sum\limits_{p \in {}^{\bullet}a''_i} (p,\, a''_i)$, $A'_{\mathrm{TE}}(b_{\mathrm{out}}) := \sum\limits_{p \in {}^{\bullet}b_{\mathrm{out}}} (p, b_{\mathrm{out}})$ that is stated in algorithm 2, for $\forall t \in T'_{\mathrm{E}}$ and $\forall p_i \in {}^{\bullet}t (i = 1, 2, \cdots, n)$, one will have $\sum\limits_{i=1}^{n} A_{\mathrm{FE}}(p_i,\, t) = A_{\mathrm{TE}}(t)$. This proves $CN'_{\mathrm{E}}$ meets the third property of well-structured component.

That concludes the proof.

## 7. Conclusion

Under the framework of component net modeling process based on EEOOPN, this paper discussed the problem of performance improvement during the process of component evolution. This objective is achieved by extending concurrency between component nets. It is using the partition block dependence analysis technique and component net reconstruction methodology that the concurrency is extended into a global manner while well-structured property of a component net is preserved.

**References**

[1] K. Jensen, L. M. Kristensen: *Colored petri nets: A graphical language for formal modeling and validation of concurrent systemst.* Communications of the ACM *58* (2015), No. 6, 61–70.

[2] N. Zhao, J. Wang, T. Li, Y. Yu, F. Dai, Z. Xie: *ESDDM: A software evolution process model based on evolution behavior interface.* International Conference Intelligent Computing and Information Science, 8–9 Januar 2011, Chongqing, China, Part II, Springer, Book Series (CCIS) *135* (2011), 562–567.

[3] A. Al-Shabibi, D. Buchs, M. Buffo, S. Chachkov, A. Chen, D. Hurzeler: *Prototyping object oriented specifications*: International Conference on Application and Theory of Petri Nets, Petri Nets and Other Models of Concurrency, 23–27 June 2003, Eindhoven, The Netherlands, Springer, Book Series (LNCS) *2679* (2003) 473–482 .

[4] S. Pashazadeh, M. Rahimi: *Modeling timestamp ordering method using colored petri net.* Indian Journal of Science & Technology *8* (2015), No. 35, paper 77633.

[5] A. Alekseyev, V. Khomenko, A. Mokhov, D. Wist, A. Yakovlev: *Improved parallel composition of labelled petri nets.* IEEE Eleventh International Conference on Application of Concurrency to System Design, 20–24 June 2011, Newcastle Upon Tyne, UK, IEEE Conference Publications (2011), 131–140.

[6] L. D. da Silva, K. Gorgonio, A. Perkusich: *Petri nets for component-based soft-*

*ware systems development*. Petri Net, Theory and Applications, Book Publisher: I-Tech Education and Publishing, Vienna, Austria (2008).

[7] T. Li, H. Yang, J. Jiang: *Mining for concurrency in software process for evolution.* 10th Joint International Computer Conference (JICC 2004), 4–6 November 2004, Kunming, China, International Academic Press, Beijing (2004) 478–483.

[8] T. Li: *An approach to modeling software evolution processes*. Springer-Verlag Berlin Heidelberg, Distribution rights in China: Tsinghua University Press (2009).

Received April 30, 2017