Advances in Information Mining
ISSN: 0975–3265 & E-ISSN: 0975–9093, Vol. 3, Issue 1, 2011, pp-33-43
Available online at http://www.bioinfo.in/contents.php?id=32

USING INCREMENTAL MULTI PERSPECTIVE PROCESS MINING ALGORITHM IN PROCESS ANALYSIS OF CSCW SYSTEMS

TAHEREH SADAT MOUSAVI[1]*, AHMAD BARAANI DASTJERDI[2]
[1]Department of Computer Engineering, Faculty of Engineering, University of SheikhBahai, Baharestan, Isfahan, Iran
[2]Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Hezar Jerib Ave., Isfahan, Iran
*Corresponding Author: tsmousavi87@gmail.com

Received: August 27, 2011; Accepted: November 15, 2011

Abstract- In CSCW systems, the end results are subject on how personnel participate and work specifically in a flexible dynamic environment. Thus, it has become imperative to develop a comprehensive system whereby the analysts can understand and analyze the workflow in order to enhance personnel support, cooperation processes and the final results. In other to achieve this goal, a system whereby individual and groups send feedbacks to the personnel are advocated. If analysts have access to the workflow and personnel information during the execution of work, they will be able to achieve the above-mentioned goals faster through analysis and sending feedback, especially in operational environments in which time is important. In this article, an Incremental Process-mining technique is presented to increase efficiency in a multi perspective process model during the execution of work. This technique is integrated within the ProM6 framework. The results obtained from the proposed algorithm which is based on criteria such as Fitness, Cost-Based Fitness etc tends to agree with alpha algorithm.
Key words- Computer Supported Cooperative Work, Process Mining, Process Mining Perspectives

1. Introduction

CSCW tries to support group work by using tools and groupware technologies taking cognizance of sociological, organizational and psychological factors [1]. In cooperative ventures, teams work together to accomplish a task [2]. In cooperative systems, things are done through team participation and cooperation, the final result depends on the way people participate and work, especially in those environments which do not follow a constant procedure. Therefore, specifying the workflow is important in realizing how the works are done. On the other hand, analyzing the workflow procedures is important in order to improve personnel support, cooperative processes and the final results. One way to achieve the latter is giving individual and group feedbacks. Individual feedbacks encourage the personnel to work and collaborate together in a better way. If analysts have access to the workflow and personnel information – process perspective and organizational perspective during analysis process, they will be able to achieve the above-mentioned goals faster through analysis and sending feedback, especially in operation environment in which time is important. Therefore, helping the analysts in analyzing and giving feedback during the analysis process would be much effective in achieving their goals.
One important feature of research about CSCW systems is its capability to give researchers' access to information logs of human-machine interactions. Whilst using logs and analyzing them in order to specify the processes within the CSCW area is of primary importance, it is one of the most challenging aspects of doing research in this field [3].
Based on works done in this area, the well-known process-mining methods have some shortcomings which are its inability to process the log during execution and represent an output model which encompasses the process perspective and also, organizational perspective information in an integrated way. In other words, they are not able to offer the needed information about the workflow and the people so that giving individual feedback becomes impossible.
In this article, an incremental multi perspective process-mining technique is presented. This technique processes the log within process and organizational perspectives and presents a model containing information about both perspectives as its output. In this way, it makes analyzing the process and giving individual and group feedback during the execution of work in CSCW systems possible. In this method, each time a new log is produced, the resulting model is compared with the existing one in the highest level – the model level – and the existing model is improved. The results of evaluating the two merged model instances are given in the proposed algorithm which are based on criteria like Fitness , Cost-Based Fitness, and ETC Precision and comparing them with the results of alpha algorithm which is a known algorithm in process-mining and examining organizational perspective as well, confirms the operation and output of the proposed algorithm. In this article, "workflow" and "process" are used equivalently.
This article is organized as follows; related works are mentioned in section 2 which is followed by the proposed

incremental multi perspective process-mining technique in section 3. Sections 4 and 5 talk about implementation and evaluation of the proposed technique and finally section 6 is dedicated to conclusion and future works.

2. Related Works

CSCW and CSCL systems are two research areas which share some research aspects like analyzing cooperative processes. CSCL systems were introduced after CSCW systems. CSCL is a new educational paradigm which is based on socially inspired theories, tries to support collaborative methods using computer science technologies [4].

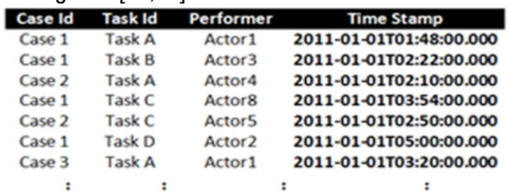
In [4], a framework for a comprehensive analysis of CSCL synchronized environments using "problem-solving" method is presented. In this framework, the analysis is semi-automatic and a high amount of work must be done by the evaluator. In [3], the Heuristic Miner algorithm is used as a heuristic algorithm[5,6] for decision-making process analysis in CSCL systems. But Heuristic Miner does the processing after the work has been completed and only also within the process perspective.

In [7], the idea of comprehensive process-mining is introduced. In [8], a multi perspective process-mining method is introduced by which a comprehensive model named simulation model is derived automatically. None of the methods are incremental.

In articles [9-12] the incremental process-mining method is presented that is specifically used to mine system and software engineering processes from the existing information in Software Configuration Management Systems, this method isn't multi perspective. In [13] another incremental process-mining method is introduced which works on the principles of intermediate relationships. This method processes the information log only from the process perspective. No incremental multi perspective process-mining method has been discussed past.

3. The Incremental Multi Perspective Process Mining Technique

The traditional approaches of process design are time consuming and prone to error. Their final result is the process to be executed instead of the real process being executed. Process-mining is one of the modern process design methods which derive real process from transaction log as shown in Fig. (1) And also, contains the execution information of the real process [14]. Therefore, process mining is a suitable approach for designing real processes in process-oriented systems. The basics of process-mining are derived from data-mining field [15,16].



Case Id	Task Id	Performer	Time Stamp
Case 1	Task A	Actor1	2011-01-01T01:48:00.000
Case 1	Task B	Actor3	2011-01-01T02:22:00.000
Case 2	Task A	Actor4	2011-01-01T02:10:00.000
Case 1	Task C	Actor8	2011-01-01T03:54:00.000
Case 2	Task C	Actor5	2011-01-01T02:50:00.000
Case 1	Task D	Actor2	2011-01-01T05:00:00.000
Case 3	Task A	Actor1	2011-01-01T03:20:00.000
⋮	⋮	⋮	⋮

Fig. 1- Transaction log 1

Process-mining techniques can process the log from different perspectives. Based on W. M. P. Van der Aalst researches [17], process-mining includes 3 basic perspectives:

a. Process-perspective: is related to the control flow; it mainly focuses on activities and relationships among them. The output of process-mining from this perspective is process model [18]. One of the common modeling languages is Petri net or P/T-net. In the classic model, a Petri net is N = (P, T, F) [19] , in which:
   - P is a finite set of places (the circles in model 1 of Fig. 5);
   - T is a finite set of transitions for which $P \cap T = \emptyset$. Each transition specifies a task (the rectangles in model 1 of Fig. 5);
   - $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs.

b. Organizational perspective: is related to organizational structure and people related to the process. Its main focus is on people and their relationships [20].

c. Data perspective: is related to process instances and the information related to them.

Multi perspective process-mining technique is a technique that is able to mine multiple perspectives and whose output is a model consisted of different perspectives information [7, 8]. The incremental process-mining technique derives the process model from ongoing execution of the main process is the process-mining technique. Moreover, any change in the main process affects the process model as well [9].

The technique which processes the log during its completion and its output model includes different perspectives information is the incremental multi perspective process-mining technique. This technique is novel in the area of process mining. An incremental multi-perspective process-mining technique is introduced. Fig. (2) Shows the overview of the technique. The technique encompasses three major phases of deriving the process model, adding the organizational perspective and comparing and merging the existing model with the new one (rectangles in Fig. (2)). First, with the creation of *Log 1*, *Model 1* is derived by *Alpha algorithm*. In the second phase, the organizational perspective is added to the process model (*Multi-perspective Model 1*) by *Multi- Perspective* algorithm. When a new log *(Log2)* is created, the phases are repeated and Model2 and *Multi-perspective Model 2* are derived. In this stage, there are two models (*Multi perspective Models* 1 and 2); these models are compared and merged by *Comparison and Merge* algorithm and *Merged Model* is derived – as presented in Fig.(2). Each time, by the execution of *Comparison and Merge* algorithm, the merged model is replaced by the existing one and whenever a new log is created, all the phases are repeated.

Logs are received through time window. Based on the operational environment, the window duration is chosen so that it contains at least one full instance of process execution. By the end of each time span a new log is received; for example in Fig. (2), log 2 is received after one window time span and after log 1 has been received. The technique is integrated within the ProM6 framework as a Plug-in. This framework is an open-source process-mining software by which it is possible to use different process-mining algorithms as plug-ins [21,22]. A detailed explanation of the phases is presented below.

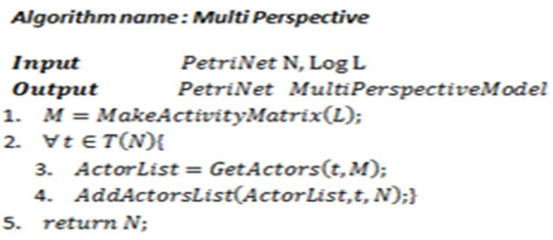
a. Derivation of The Process Model

Process is derived from log through the Alpha algorithm [18]. The Alpha algorithm receives the information log as input and after identifying the causality relations it gives the process, which has been modeled based on Petri Net.

As mentioned before, in each phase, log is received after one time window. Based on the operational environment, the length of time window is chosen so that it contains at least one full instance of process execution. By receiving the log and by using the alpha algorithm, the process model is derived for the new log and then is compared and merged with the existing model. Because the process model is derived using the alpha algorithm, it is tried to consider the way the alpha algorithm works throughout the compare and merge algorithm so that the final model is complete and accurate.

b. Adding The Organizational Perspective to The Model

As mentioned earlier, the purpose on the incremental multi perspective process-mining technique is to make individual feedback and more detailed analyses possible. For this reason, in this phase, people are added to the model from organizational perspective. This is done in a way that the analyzer is able to recognize who has done each activity and how many times. Fig. (3) represents the multi perspective algorithm pseudo code.

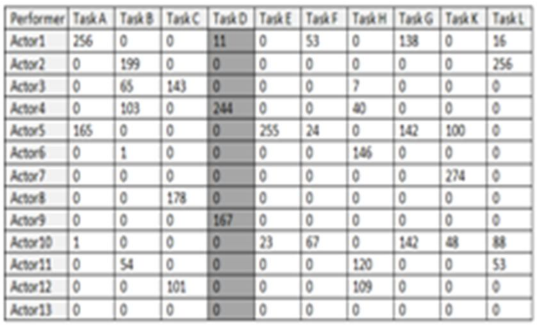


```
Algorithm name : Multi Perspective

Input       PetriNet N, Log L
Output      PetriNet MultiPerspectiveModel
1. $M = MakeActivityMatrix(L);$
2. $\forall t \in T(N)\{$
    3. $ActorList = GetActors(t, M);$
    4. $AddActorsList(ActorList, t, N);\}$
5. return N;
```

Fig. 3- The Multi perspective algorithm pseudo code

To add people information to the model, first an activity matrix [9] is created like the one in Fig. (4)(line 1 of Fig. (3)). In activity matrix, rows show the individuals and columns portrays the activities and numbers show the number of executing an activity by a certain actor. For example, M[2][5] shows that Actor1 executed TaskD 11 times. Afterwards, based on the Joint Activity metric [20] (line 2 of Fig. (3)),), the individuals' information is derived from activity matrix (line 3 of Fig. (3)) and is added to the model (line 4 of Fig. (3)).

Fig. (5) shows an instance of a multi perspective model. For example TaskD, which is highlighted in the matrix of Fig. (4), is done 11 times by Actor1, 244 times by Actor4 and 167 times by Actor9. As exhibited in Fig. (5), this information is shown by clicking the grey circle connected to TaskD.



Performer	Task A	Task B	Task C	Task D	Task E	Task F	Task H	Task G	Task K	Task L
Actor1	256	0	0	11	0	53	0	138	0	16
Actor2	0	199	0	0	0	0	0	0	0	256
Actor3	0	65	143	0	0	0	7	0	0	0
Actor4	0	103	0	244	0	0	40	0	0	0
Actor5	165	0	0	0	255	24	0	142	100	0
Actor6	0	1	0	0	0	0	146	0	0	0
Actor7	0	0	0	0	0	0	0	0	274	0
Actor8	0	0	178	0	0	0	0	0	0	0
Actor9	0	0	0	167	0	0	0	0	0	0
Actor10	1	0	0	0	23	67	0	142	48	88
Actor11	0	54	0	0	0	0	120	0	0	53
Actor12	0	0	101	0	0	0	109	0	0	0
Actor13	0	0	0	0	0	0	0	0	0	0

Fig. 4- The activity matrix for Log 1

c. Comparing and Merging The Existing and The New Model

Based on the purpose of the research, the existing model must be completed whenever a new log is received. To do this, the new model must be compared with the existing one. The purpose of comparison in the comparison and merge algorithm is to answer the following question: "does the existing model include the new model or not?"

If the existing model includes the new one, it will be enough to update and edit the organizational perspective; if not, the existing model must be changed so that it includes the new one.

Considering that in process-mining the process is basically a series of events and in Petri Net modeling language tasks are the main parts of the model and because of their importance in CSCW systems, in this research tasks and their execution traces are considered as the main criteria in comparing the models. In other words, in order to compare the existing model with the new one, with the aforementioned goal, tasks and their execution traces are compared.

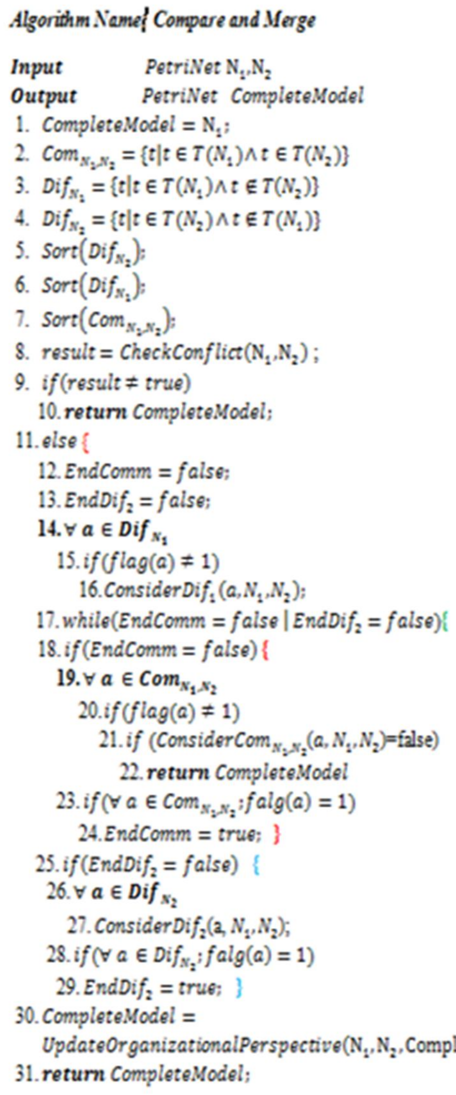
In as much as the comparison is done at model level and on the basis of what is said in [3], the relationships among tasks based on the model and by a few changes for Petri Net like $N = (P, T, F)$, are defined as follows:

1. $t_1 \underset{N}{\rightarrow} t_2$ (ordering): if there is a trace $\{x_1, x_2, \dots, x_n\}$ so that for $1 \leq i < n$, $(x_i, x_{i+1}) \in F(N)$ and $x_1 = t_1$ and $x_n = t_2$.
2. $t_1 >_N t_2$ (sequence): if there is a trace $\{x_1, x_2, \dots, x_n\}$ so that for $1 \leq i < n$, $(x_i, x_{i+1}) \in F(N)$ and $x_i = t_1$ and $x_{i+1} = t_2$. Sequence is a special case of ordering.
3. $t_1 \wedge_N t_2$ (parallel): if there are two trace $\{x_1, x_2, \dots, x_n\}$ and $\{y_1, y_2, \dots, y_m\}$ so that for $1 \leq i < n, (x_i, x_{i+1}) \in F(N)$ and for $1 \leq j < m$, $(y_j, y_{j+1}) \in F(N$ and $\{y_2, \dots, y_m\} \cap \{x_2, \dots, x_n\} = \emptyset$ and $x_1 = y_1 \in T(M)$ and $x_n = t_1$ and $y_m = t_2$.
4. $t_1 \vee_N t_2$ (exclusive): if there are two traces $\{x_1, x_2, \dots, x_n\}$ and $\{y_1, y_2, \dots, y_m\}$ so that for $1 \leq i < n$, $(x_i, x_{i+1}) \in F(N)$ and for $1 \leq j < m$, $(y_j, y_{j+1}) \in F(N)$ and $\{y_2, \dots, y_m\} \cap \{x_2, \dots, x_n\} = \emptyset$ and $x_1 = y_1 \in P(N)$ and $x_n = t_1$ and $y_m = t_2$.

The pseudo code for the comparison and merge algorithm is shown in Fig. (6).

To compare and merge the existing model (N_1) and the new model (N_2), tasks are divided into three sets of Com_{N_1,N_2}, Dif_{N_1} and Dif_{N_2} and to reduce the number of comparisons and examinations, tasks are ordered based on their occurrence in the model through the Bubble sort method (lines 2-7 of Fig. (6)).

The new model must be similar to the existing model in some properties. However, the existing model and the new one can be merged when they do not have conceptual contrast. For this reason, the two models are first examined by the *Check Conflict* algorithm to see whether they can be merged; if they can, the algorithm continues and if not, the existing model is returned unchanged (lines 8-10 of Fig. (6)). when algorithm is to be executed first, Dif_{N_1} set is examined by $\mathrm{ConsiderDif}_{\mathrm{N1}}$ algorithm (lines 14-16 of Fig. (6)) and then, Com_{N_1,N_2} and Dif_{N_2} sets are examined by $Consider\boldsymbol{Com}_{\boldsymbol{N_1,N_2}}$ and *ConsiderDif*$_{N_2}$ algorithms (lines 17-29 of Fig. (6)).



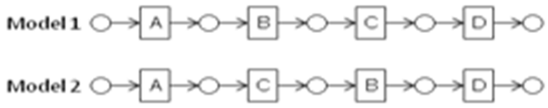
```
Algorithm Name{ Compare and Merge

Input       PetriNet N_1,N_2
Output      PetriNet CompleteModel
1. CompleteModel = N_1;
2. $Com_{N_1,N_2} = \{t | t \in T(N_1) \wedge t \in T(N_2)\}$
3. $Dif_{N_1} = \{t | t \in T(N_1) \wedge t \in T(N_2)\}$
4. $Dif_{N_2} = \{t | t \in T(N_2) \wedge t \notin T(N_1)\}$
5. $Sort(Dif_{N_2})$;
6. $Sort(Dif_{N_1})$;
7. $Sort(Com_{N_1,N_2})$;
8. result = CheckConflict(N_1,N_2) ;
9. if(result ≠ true)
   10. return CompleteModel;
11. else {
   12. EndComm = false;
   13. $EndDif_2$ = false;
   14. $\forall a \in Dif_{N_1}$
     15. if(flag(a) ≠ 1)
       16. $ConsiderDif_1(a, N_1, N_2)$;
   17. while(EndComm = false | $EndDif_2$ = false){
   18. if(EndComm = false) {
     19. $\forall a \in Com_{N_1,N_2}$
       20. if(flag(a) ≠ 1)
         21. if ($ConsiderCom_{N_1,N_2}(a, N_1, N_2)$=false)
           22. return CompleteModel
     23. if($\forall a \in Com_{N_1,N_2}$ : falg(a) = 1)
       24. EndComm = true; }
   25. if($EndDif_2$ = false)  {
    26. $\forall a \in Dif_{N_2}$
     27. $ConsiderDif_2(a, N_1, N_2)$;
    28. if($\forall a \in Dif_{N_2}$ : falg(a) = 1)
     29. $EndDif_2$ = true;  }
30. CompleteModel =
   UpdateOrganizationalPerspective(N_1,N_2,CompleteModel);
31. return CompleteModel;
```

Fig. 6- The Comparison and Merge algorithm pseudo code

Based on the above mentioned relations, the two models aren't similar and cannot be merged under two criteria:

At the end, the organizational perspective of the merged model is updated by *Update Organizational Perspective* algorithm and the Multi-perspective merged model is returned (lines 30-31 of Fig. (6)).

If the order of task execution are different in the two models. Based on the fact that the order of tasks specify the causality relations, difference in task order in two models shows that the models are not similar which indicates conflict. For example in model 1 of Fig. (7) the execution of C depends on the execution of B and the execution of D depends on the execution of C, while in model2 the execution of B depends on the execution of D and the execution of C depends on the execution of B. This shows that the two models cannot be merged. Such examination is done by $\mathrm{ConsiderCom}_{\mathrm{N_1,N_2}}$ algorithm



` Fig. 7- Process models in Petri Net

1. If , $C \in N_1$, $B, C \in N_2$ and $B \wedge_{N_1} C$ then there has to be $B \wedge_{N_2} C$ or at least $B >_{N_2} C$ or $C >_{N_2} B$ so that it is possible to make them parallel; if not, models 1 and 2 cannot be merged.

It is noticeable that if $B, C \in N_1$ and $B \wedge_{N_1} C$ but $B, C \notin N_2$ or only one of them belongs to the new model, then there is no conflict (Fig. (8)).

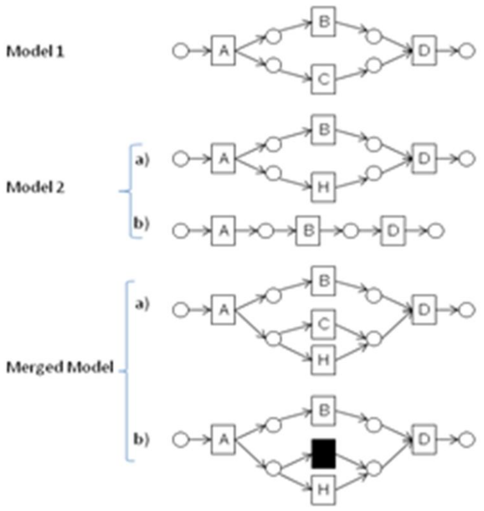


Fig. 8- Process models in Petri Net

This similarity condition is put in Check Conflict algorithm like the pseudo code in Fig. (9) below.

```
Algorithm Name: Check Conflict

Input      PetriNet N_1,N_2

Output     Boolean

1.$AND_{N_1} = \{(a,b) | a \in T(N_1) \wedge b \in T(N_1) \wedge a \wedge_{N_1} b\}$
2.$AND_{N_2} = \{(a,b) | a \in T(N_2) \wedge b \in T(N_2) \wedge a \wedge_{N_2} b\}$
3.$\forall (a,b) \in AND_{N_1}$
  4.$if(a \in Com_{N_1,N_2} \wedge b \in Com_{N_1,N_2})$
    5.$if((a,b) \notin AND_{N_2})$
      6.$if(!\,CheckParallelSituation(a,b,N_2))$
        7.$return\ false;$
8.$\forall (a,b) \in AND_{N_2}$
  9. $if(a \in Com_{N_1,N_2} \wedge b \in Com_{N_1,N_2})$
    10.$if((a,b) \notin AND_{N_1})$
      11.$if(!CheckParallelSituation(a,b,N_1))$
        12.$return\ false;$
```

Fig. 9- Check Conflict algorithm pseudo code

After checking the possibility of merging the models, members of the three aforementioned sets are examined. Dif_{N_1} consists of those tasks which are in the existing model but not in the new one. While these tasks exist in the basic model, they are in their right place but if not, task is replaced. Thus, prompting a need to make changes in the existing model (adding invisible task) like Fig. (10).

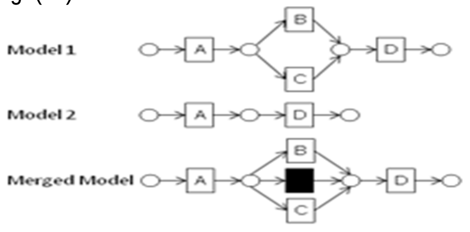


Fig. 10- Process models in Petri Net

Fig.(11) shows the pseudo code for $\mathrm{ConsiderDif}_{N1}$ algorithm. $flage(a) \neq 1$ i.e. a is not considered.

By considering points before and after each task in Dif_{N_1} and finding the corresponding conditions in model 2, this algorithm checks if any task is being done between points before and after the specified task in model 2. If there is no task, a hidden task with an exclusive relation to the specified task (Fig. (10)) is added to the model (lines 12, 14 and 29 of Fig. (11)).

While anything in $\boldsymbol{Com_{N_1,N_2}}$ is also in both models, models 1 and 2 has to be compared for each member of this set. As mentioned before, the purpose of this comparison is to answer this question: "does the existing model include the new model or not?"

There are several ways to compare two models the output of which is mostly true/false [24]. The method employed in this research is like Trace/String Equivalence method [23-25] in using executive traces to check difference between models. Based on its definition, in Trace Equivalence method two models are similar if each executive trace in model1 exists in model 2 and each executive trace in model2 exists in model1. The output is true/false. The equivalence method is criticized for:

1. Infinite number of traces
2. Inability to recognize the correct selection point location

```
Algorithm Name : ConsiderDif_1

Input      Transition a, PetriNet N_1,N_2

Output     Void

1. $\forall (a,b) \in AND_{N_1}$ {
2.$if(a \in Dif_{N_1} \wedge b \in Dif_{N_1})$
3. $if(flage(a) \neq 1 \wedge flage(b) \neq 1)$ {
  4. $BT = beforeTransition(a,b,N_1);$
  5. $AT = AfterTransition(a,b,N_1);$
  6. $if(getBeetwenTransitions(BT,AT,N_2) = null)$
    7. $MakeParallel((a,b),nothing,N_1);$}
8. $if(a \in Com_{N_1,N_2} \wedge b \in Dif_{N_1})$
 9. $if(flage(b) \neq 1)$
  10.$if(\exists\, (a',b') \in AND_{N_2} : a' = a)$
    11.$if(b' \in Dif_{N_2})$
      12.$MakeExclusive(b,b',N_1);$
  13.$else$
    14.$MakeExclusive(b,nothing,N_1)$}
15.$\forall a \in Dif_{N_1}$
 16.$if(flag(a) \neq 1)$ {
  17.$BT = BeforeTransition(a,N_1);$
  18.$if(BT \in Dif_{N_2})$
    19.$while(BT \in Dif_{N_2})$ {
      20.$BT = BeforeTransition(BT,N_1);$
      21.$flag(BT) = 1;$}
  22.$AT = AfterTransition(a,N_1);$
  23.$if(AT \in Dif_{N_2})$
    24.$while(AT \in Dif_{N_2})$ {
      25.$AT = AfterTransition(AT,N_1);$
      26.$flag(AT) = 1;$}
  27.$if(getBeetwenTransitions(BT,AT,N_2) = null)$
    28.$if(AT \notin AfterTransitions(BT,V))$
      29.$MakeExclusive(getBeetwenTransitions(BT,AT,C),nothing,N_1);$}
```

Fig. 11- $\mathrm{ConsiderDif}_{N1}$ algorithm pseudo code

But in the present work, as mentioned earlier, equality is not under study; on the one hand, based on the assumption that in this method there is no loop, the first problem will not occur and on the other hand, because of the special conditions of the case under study in this research, the goal and the type of comparison, the exact recognition of selection points is not important very much. In addition, disregarding the exact recognition of a number of selection points, which rarely occur, will prevent the problem of over-fitting [26].

To examine each task in $\boldsymbol{Com_{N_1,N_2}}$, all executive traces from the beginning of the process to the specific task are extracted and compared from both models. If the set of new model traces for the specified task are a subset of the set of existing model traces for the same task, then the existing model includes all the possible states for the task in the new model; if not, all transferable changes are made in the existing model for every task. The consider$\boldsymbol{Com_{N_1,N_2}}$ algorithm pseudo code is shown in Fig. (12).

```
Algorithm name : $ConsiderCom_{N_1,N_2}$

Input        Transition a,
             PetriNet $N_1, N_2$
Output       Boolean
1. $Traces_{N_1}(a) = N_1.GetTraces(start, a);$
2. $Traces_{N_2}(a) = N_2.GetTraces(start, a);$
3. $\forall\ trace_i \in Traces_{N_2}(a)\ ;\ 1 \le i \le |Traces_{N_1}(a)|\{$
  4. $if\ (trace_i \in Traces_{N_1}(a))$
    5. $if(checkParallelSituation(a, N_1))$
      6. $Change(a, N_1);$
    7. $else\ if(EndDif_2 = true)$
      8. return false;}
9. return true;
```

Fig. 12- The Consider$\boldsymbol{Com_{N_1,N_2}}$ algorithm pseudo-code

Because of the examination of parallel states in *Check Conflict* and *Consider*Dif_{N_1} algorithms, there is only one allowed status remained for non-equality of executive traces in *Consider*$\boldsymbol{Com_{N_1,N_2}}$ which is shown in Fig. (13).

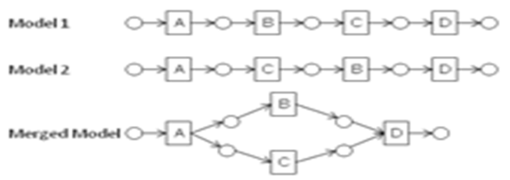


Fig. 13- Process instances in Petri Net

The tasks in Dif_{N_2} do not exist in the existing model and thus, they have to be placed in their suitable position – the same as their position in the new model – in the existing model (Fig. (14)).

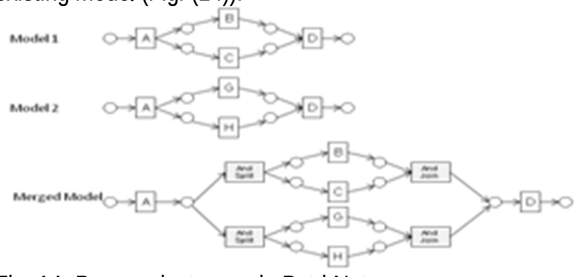


Fig. 14- Process instances in Petri Net

Because of the importance of task order for causality relations, in the process of adding a task, it must be added in a correct position to the model. in this regard, it is important to answer the following questions:

1. After what other task(s) is a specific task executable?
2. Can any other task be executed during the execution of a specific task?
3. What other task(s) can be executed after a specific task?

Based on what mentioned, the ConsiderDif_{N_2} algorithm is shown in Fig. (15).

By examining the three sets mentioned, all tasks in both models are considered and the process perspective is studied. Now is time to work on organizational perspective.

The merged model should include all the information related to the organizational perspective of both models. Based on this account, the organizational perspective information is studied for every task in Com_{N_1,N_2} and Dif_{N_2} task sets and then added to the existing model. At the end a merged multi perspective model is returned. The pseudo-code for considering organizational perspective is shown in Fig. (16).

```
Algorithm Name : Complete Organizational Perspective
Input        PetriNet $N_1, N_2$

Output       PetriNet $N_1$

1. $\forall t \in Dif_{N_2}$
  2. $InsertActorsList(t.ActorsList, t, N_1);$
3. $\forall t \in Com_{N_1,N_2}\{$
  4. $AcorsList = AddActorsList(t, N_2, N_1);$
  5. $InsertActorsList(ActorsList, t, N_1);\}$
6. return $N_1$;
```

Fig. 16- Updating Organizational Perspective algorithm pseudo code

The merged model as described section 1 exhibited some improvement because it is able to analyze the process and give individual and group feedbacks by presenting the process model and people involved in it (personnel) for the analysts.

4. Implementing the Incremental Multi Perspective Process Mining Algorithm

The Incremental Multi Perspective Process Mining algorithm, which was explained in section 3, is integrated within ProM6 framework. Considering unavailability of an instance of a CSCW system and its event log to check the algorithm's operation, so a log has been designed for this reason.

Considering that the purpose is to make the existing model complete, the information log has been designed so that it can represent part of the algorithm's capabilities. In order to provide the necessary input for the algorithm, the log must contain the following information (Fig. (1)): 1) process instance number or case Id, 2) Task/Activity Id, 3) time, and 4) performers or personnel. Total log consists of 8 different traces and 739 trace instances.

As was shown in the overview of the incremental multi perspective process-mining technique (Fig. (2)), in this process-mining technique, the log is received gradually during several phases through time windows. In this respect and also in order to show the algorithm's performance, the log is divided into three parts and is received in three phases (Fig. (17)).

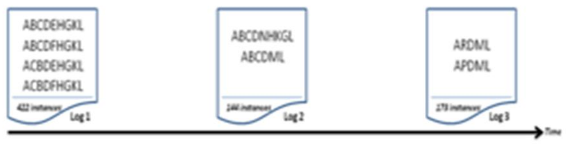


Fig. 17- Receiving the log in three phases

Below, the technique's operation is explained phase by phase by receiving the log.

- **Receiving Log 1**

By receiving Log 1, the process model 1 is derived by alpha algorithm based on what was explained in section 3.1. This is followed by deriving the multi perspective model 1 through the multi perspective algorithm (section 3.2., Fig. (3)) like as Fig. (5).

- **Receiving Log 2**

After one time window, Log 2 is received and alpha and multi perspective algorithms are used for this log again and multi perspective model 2 is made (Fig. (18)).

After deriving the new model from the new log (Log 2), there exist two models: model 1 and model 2. The two models are test run in other to compare and merge algorithm to reach to a more complete model. The result of comparing and merging is the multi perspective model 2-1 (Fig. (19)).

As is obvious, in comparing and merging:

$$Com_{N_1,N_2} = \begin{Bmatrix} \text{TaskA, TaskB, TaskC, TaskD, TaskH, T}as\text{kK,} \\ \text{TaskG, TaskL} \end{Bmatrix}$$

$$Dif_{N_1} = \{\text{TaskE, TaskF}\}$$

$$Dif_{N_2} = \{\text{TaskM, TaskN}\}$$

- Considering that $TaskB \wedge_{N_1} TaskC$ and $TaskB >_{N_2} TaskC$, based on check-conflict algorithm (fig. 10) models 1 and 2 are similar, there is no conflict and they can be merged.
- In considerDif_{N_1} , because instead of TaskE and TaskF in model 1, TaskN is performed in model 2, there is no need to apply changes in execution of considerDif_{N_1}.
- Members of Com_{N_1,N_2} set are checked in two stages. Considering that trace sets TaskC, TaskB, TaskA and TaskD in model 2 are subsets of their trace sets in model 1, checking is finished. But for TaskH (also for TaskK, TaskG and TaskL), so they will be considered in next step.

  $$Traces_{N_1}(H) = \{ABCDEH, ABCDFH, ACBDEHACBDFH\}$$

  $$Traces_{N_2}(H) = \{\text{ABCDNH}\}$$

  $$Traces_{N_2}(H) \not\subseteq Traces_{N_1}(H)$$

- In considerDif_{N_2}, TaskN and TaskM are added to model 1 in their proper location. Afterwards, in reconsidering the set Com_{N_1,N_2} , all members are checked and there would be no problem. For example for TaskH:

$$Traces_{N_1}(H) \begin{Bmatrix} ABCDEH, ABCDFH\ , ACBDEH \\ ACBDFH, \text{ABCDNH,} \\ \text{ACBDNH} \end{Bmatrix}$$

$$Traces_{N_2}(H) = \{\text{ABCDNH}\}$$

$$Traces_{N_2}(H) \subseteq Traces_{N_1}(H)$$

In checking TaskG and TaskK, noticing their different ordering in the two models, their parallel execution is specified and in the merged model their parallel status is shown.

- After checking all three sets, the organizational perspective is reedited based on the common activity criterion at the model level and the merged model will contain the total organizational perspective of both models. This is shown in Fig. (5, 18 and 19) for TaskD.

- **Receiving Log 3**

After the second time window, Log 3 is received and alpha and multi perspective algorithms are executed for this log as well and multi perspective model 3 is made (Fig. (20)).

After deriving the new model from the new log (Log 3), two models are obtained which are: the existing model (model 1-2) and the new model (model 3). The compare and merge algorithm is used again in order to make a more complete model. The result is the multi perspective model 1-2-3 (Fig. (21)).

5. Evaluating The Incremental Multi Perspective Algorithm

In evaluating the proposed algorithm, a new approach has been devised based on the fact that there is no reference algorithm for comparison. For this reason, checking the performance of the algorithm and the accuracy (or correctness) of its output is presented in two sections below. In 5.1. the output accuracy of process perspective and in 5.2. The output accuracy of organizational perspective is considered.

5.1.Evaluating Process Perspective

It should be recalled that in the first phase of the incremental multi perspective process-mining algorithm, the alpha algorithm is used to derive the process model. In checking the process perspective the models, which are made by merging the existing and the new model – the incremental models – are compared with the model derived directly from the alpha algorithm.

Based on the accessibility to log, the use of Petri net models and also no access to the basic process model, fitness, Cost-based fitness and Precision [27-31] are used for the evaluation.

The fitness criterion specifies how much behavior observed in log is parsed by the model [27-29]. In Cost-based Conformance, by considering the cost of skipped and inserted activities, which cause deviations in the model and model behaviors different from what is observed in the log, the value of Cost-based fitness is calculated [30]. The Precision criterion refers to overly general models, preferring models with minimal behavior to represent as closely as possible the log.

In order to apply precision the method in [31] is used. In this method, Precision is calculated by determining Escaping Edges and their frequency.

Precision, Cost-Based Fitness and Fitness are calculated through Compute Fitness Plugin, *Reply a Log on Petri*

Net Conformance Analysis Plugin and *Check Conformance using ETConformance* plugin respectively that exist in ProM6 framework.

Fig 20 showed that the output model of alpha algorithm when its input log contains the information in log 1 and log 2 is similar to the process model regardless of its organizational perspective.

Table 1 and 2 show Fitness , Cost-Base Fitness and ETC Precision values for process models derived through alpha algorithm and the incremental multi perspective algorithms.

Table-1 - evaluating the output process models of alpha and the incremental multi perspective process-mining algorithms (the input log = log 1 + log 2)

Metrics		Output of Alpha Algorithm	Output of Incremental Multi Perspective Algorithm
Fitness [0,1]		1.0	1.0
Cost-Base Fitness[0,1]	Min	1.000	1.000
	Average	1.000	1.000
	Max	1.000	1.000
ETC Precision [0,1]		0.9005	0.9005

Table-2 - evaluating the output process models of the alpha and the incremental multi perspective process-mining algorithms (the input log = log 1 + log 2 + log 3)

Metrics		Output of Alpha Algorithm	Output of Incremental Multi Perspective Algorithm
Fitness [0,1]		1.0	1.0
Cost-Base Fitness[0,1]	Min	1.000	1.000
	Average	1.000	1.000
	Max	1.000	1.000
ETC Precision [0,1]		0.8802	0.8802

$Fitness = 1$ in both methods shows that models cover the log completely and there is no trace in the log that cannot be produced by the model. $Cost\_Based\ Fitness = 1$ shows that there are no skipped and inserted activities in the model and the log. $ETC\ Precision < 1$ indicates that the model is able to produce executive traces that do not exist in the log. For example the model of Fig. 20 or output model of alpha algorithm can produce traces like ACBDNHKGL, ACBDML, ACBFHKGL, etc. while they are not in the log.

The values of table (2) confirm those of table (1). The decrease in ETC Precision value in the table (2) is because of the increase in the number of traces that could be produced by the model that do not exist in the log; in other words, it is because of the increase in Escaping Edges.

The equality of evaluation criteria values of the incremental multi perspective algorithm and those of the alpha algorithm as a well-known algorithm in process-mining, confirms the accuracy of both the operation and the output of the proposed algorithm.

5.2. Evaluating The Organizational Perspective

In evaluating the accuracy of the information added to the model from organizational perspective, it is necessary to check if the set of all the performers of a specific task like TaskD in the merged model is equal to the total of sets of performers of that task in the new and the existing model. considering the visibility of performer sets of TaskD in the presented models, Table 3 shows the number of performing TaskD by different performers. According to the table, Actor1 has performed TaskD 11 times in multi perspective model 1 and 40 times in multi perspective model 2. Totally, Actor1 has performed TaskD 51 times which is equal to the desired value.

Table-3 - The number of performing TaskD by the actors

Actors / Models	Actor1	Actor4	Actor9	Sum
Multi perspective model 1	11	244	167	422
Multi perspective model 2	40	0	104	144
Sum	51	244	271	566
Multi perspective model 1-2	51	244	271	566
Multi perspective model 3	33	57	83	173
Sum	81	301	354	739
Multi perspective model 1-2-3	81	301	354	739

Therefore, based on the table, the organizational perspective information is added correctly to the merged model in each phase of merging the existing and the new model.

6. Conclusion and Future Works

In this article, the incremental multi perspective process-mining algorithm was proposed. The purpose of this algorithm was to provide suitable conditions for CSCW systems analysts to have access to the workflow and personnel information – the process perspective and the organizational perspective – during the execution of work, in order to be able to enhance work processes and results through analysis and giving feedbacks. The algorithm was implemented within ProM6 framework. Due to inability of not having access to the log of the real CSCW system, a log was designed to check the operation and outputs of the proposed algorithm and two instances of the merged model – the output of the proposed algorithm – were examined. The new log is unique in the sense that the derived multi perspective model is merged with the existing one and, the merged model contains the personnel information, the algorithm provides the necessary conditions for the analysts to analyze in a better, more comprehensive way and to be able to enhance the workflow more quickly.

The results of evaluating the output of the proposed algorithm based on criteria like Fitness, Cost-Based Fitness and ETC Precision and comparing them with the output of the alpha algorithm as a known algorithm in process-mining and, considering organizational perspective as well, confirmed the accuracy of performance and outputs of the proposed algorithm.

Future efforts can include: 1) developing the algorithm so that it will be able to deal with processes with loops; 2) Adding other process-mining perspectives which can lead the analysts to more comprehensive analysis conditions; and 3) applying the above mentioned method to other modeling languages.

7. References

[1] Grudin J. (1994) *IEEE Computer* 27(5), 19-25.

[2] Bannon L. and Schmidt K. (1989) *In Proc. First European Conf. on CSCW, Gatwick, UK. (Reprinted in J. Bowers & S. Benford (Eds.) Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Amsterdam: North-Holland*. 3-16.

[3] Reimann P., Frerejean J. and Thompson K. **(**2009**)** 1 , 98-107.

[4] Bravo C., Redondo M. A., Verdejo M. F. and Ortega M. (2008) 66, 812-832.

[5] Weijters A. J. M. M., Ribeiro J. T. S. (2011) *CIDM 2*011: 310-317.

[6] Weijters A., van der Aalst W., de Medeiros A. (2006) *WP*166, 1-34.

[7] Rembert A. J. (2006) *44th ACM Southeast Regional Conference,* 222 - 227.

[8] Rozinat A., Mans R.S., Song M. and vanderAalst W.M.P. (2009) 34, 305–327.

[9] Kindler E., Rubin V and Schäfer W. (2006) *BPMDS* 2006.

[10] Kindler E., Rubin V and Schäfer W. (2006) *Heidelberg: Springer Berlin*, 3840, 287-301.

[11] van der Aalst W.M.P., Rubin V., Verbeek H.M.W., van Dongen B., Kindler E. and G¨unther C.W. (2006) *BPM CenterReport BPM-06-30, BPM Center, BPMcenter.org*

[12] Vladimir Rubin, Christian W. Günther, Wil M. P. van der Aalst, Ekkart Kindler, Boudewijn F. van Dongen, Wilhelm Schäfer (2007) *ICSP* 2007: 169-181.

[13] Sun W., Li T., Peng W., and Sun T. (2007) 12(1), 45-55.

[14] Van der Aalst W.M.P., van Dongen B.F., Herbst J., Maruster L., Schimm G. and Weijters A.J.M.M. (2003) 47,237–267.

[15] Tiwari A., Turner C.J. and Majeed B. (2008) 14(1), 5-22.

[16] Van der Aalst W.M.P. and Weijters A.J.M.M. (2004) 53(3), 231-244.

[17] W. M.P. van der Aalst (2007) 21, 191-199.

[18] W.M.P. van der Aalst, A.J.M.M. Weijters and L. Maruster (2004) 16(9), 1128- 1142.

[19] T. Murata (1989) *Proceedings of the IEEE*, 77(4), 541-580.

[20] W. M. P. van der Aalst, H.A. Reijers and M. Song (2005) *Discovering Social Networks from Event Logs,* 14(6), 549-593.

[21] Van Dongen B., Alves de Medeiros A.K., Verbeek H.M.W., Weijters A.J.M.M. and van der Aalst W.M.P. (2005) *Springer-Verlag, Berlin*, 3536, 444–454.

[22] http://www.processmining.org/prom/downloads

[23] [23] L. Pomello, G. Rozenberg, and C. Simone (1992)*In G. Rozenberg, editor*,609, 420-472.

[24] Van der Aalst W.M.P., Alves de Medeiros A.K., and Weijters A.J.M.M. (2006) *BPM* 2006, 4102, 129-144.

[25] R. Cleaveland and S. Smolka (1999) *in J.G. Webster, editor, Encyclopedia of Electrical Engineering, John Wiley & Sons (Chap. 1 ~ 3)*

[26] Van der Aalst W.M.P. (2008) *In W. Bridewell, T. Calders, A.K. de Medeiros, S. Kramer, M. Pechenizkiy, and L. Todorovski, editors, Proceedings of the ECML-PKDD Workshop on Induction of Process Models (IPM08*), 1-2.

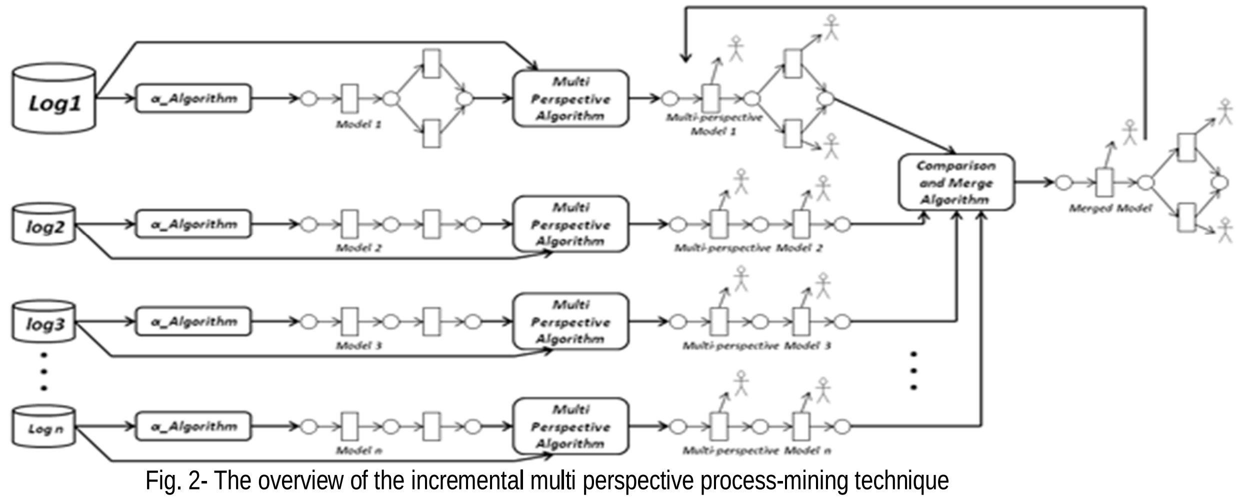
[27] Rozinat A., Alves de Medeiros A.K., Günther C.W., Weijters A.J.M.M. and van der Aalst W.M.P. (2007) *In M. Castellanos, J. Mendling, and B. Weber, editors, Informal Proceedings of the International Workshop on Business Process Intelligence* (BPI 2007),73-78.

[28] Rozinat A. and van der Aalst W.M.P. (2006) 3812, 163-176.

[29] Rozinat A. and van der Aalst W.M.P. (2008) 33(1), 64-95.

[30] Adriansyah, A., Dongen, B.F. van & Aalst, W.M.P. van der (2011) *In BPM Center Report BPM-11-11, BPMcenter.org.*

[31] Muñoz-Gama J., Carmona J. (2010) BPM 2010, 211-226



Log1
α_Algorithm
Model 1
Multi Perspective Algorithm
Multi-perspective Model 1
log2
α_Algorithm
Model 2
Multi Perspective Algorithm
Multi-perspective Model 2
Comparison and Merge Algorithm
Merged Model
log3
α_Algorithm
Model 3
Multi Perspective Algorithm
Multi-perspective Model 3
Log n
α_Algorithm
Model n
Multi Perspective Algorithm
Multi-perspective Model n

Fig. 2- The overview of the incremental multi perspective process-mining technique

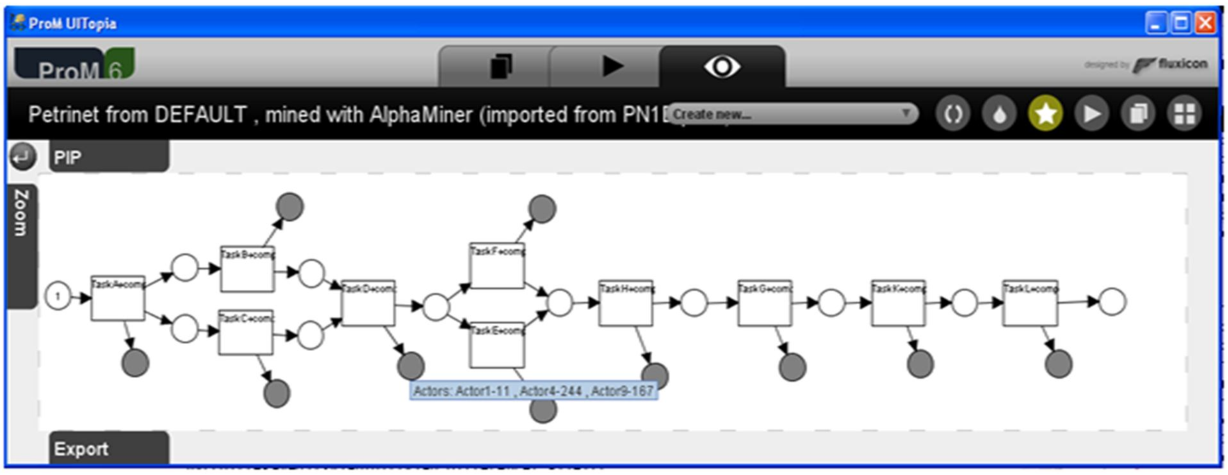
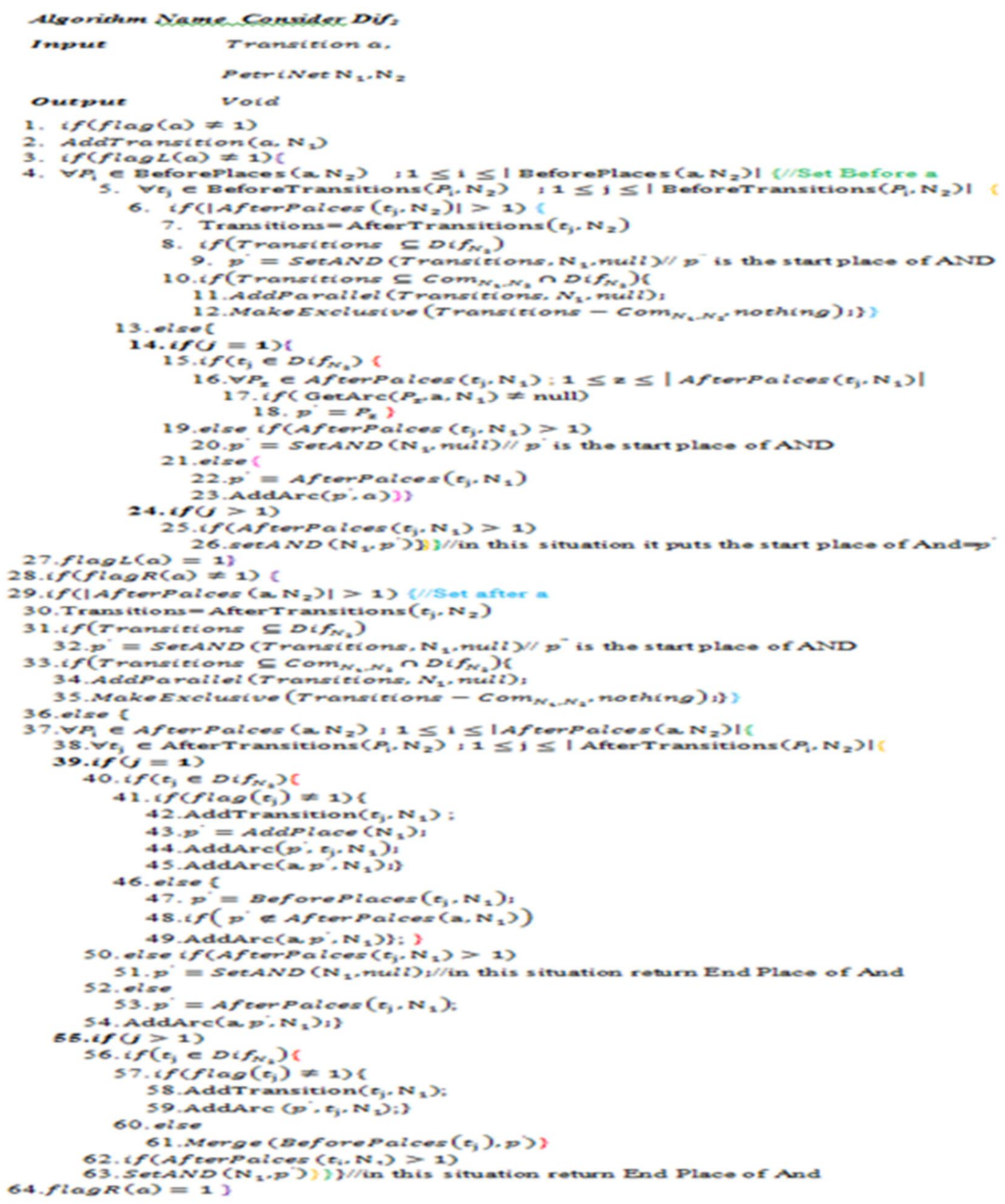


Fig. 5- Multi perspective Model 1



```
Algorithm Name  Consider Dif2
Input           Transition a,
                PetriNet N1,N2
Output          Void
1.  if(flag(a) ≠ 1)
2.  AddTransition(a, N1)
3.  if(flagL(a) ≠ 1){
4.  ∀P_i ∈ BeforePlaces(a,N2)   ;1 ≤ i ≤ | BeforePlaces(a,N2)| {//Set Before a
          5.  ∀t_j ∈ BeforeTransitions(P_i,N2)   ;1 ≤ j ≤ | BeforeTransitions(P_i,N2)| {
            6.  if(|AfterPalces(t_j,N2)| > 1) {
              7.  Transitions= AfterTransitions(t_j,N2)
              8.  if(Transitions ⊆ Dif_N1)
                9.  p' = SetAND(Transitions,N1,null)// p' is the start place of AND
              10.if(Transitions ⊆ Com_N1,N2 ∩ Dif_N1){
                11.AddParallel(Transitions, N1,null);
                12.MakeExclusive(Transitions − Com_N1,N2,nothing);}}
          13.else{
           14.if(j = 1){
              15.if(t_j ∈ Dif_N1) {
                16.∀P_z ∈ AfterPalces(t_j,N1) ; 1 ≤ z ≤ | AfterPalces(t_j,N1)|
                  17.if( GetArc(P_z,a,N1) ≠ null)
                    18. p' = P_z }
              19.else if(AfterPalces(t_j,N1) > 1)
                20.p' = SetAND(N1,null)// p' is the start place of AND
              21.else{
                22.p' = AfterPalces(t_j,N1)
                23.AddArc(p',a)}}
           24.if(j > 1)
              25.if(AfterPalces(t_j,N1) > 1)
                26.setAND(N1,p')}}}//in this situation it puts the start place of And=p'
  27.flagL(a) = 1}
28.if(flagR(a) ≠ 1) {
29.if(|AfterPalces(a,N2)| > 1) {//Set after a
  30.Transitions= AfterTransitions(t_j,N2)
  31.if(Transitions ⊆ Dif_N1)
    32.p' = SetAND(Transitions,N1,null)// p' is the start place of AND
  33.if(Transitions ⊆ Com_N1,N2 ∩ Dif_N1){
    34.AddParallel(Transitions, N1,null);
    35.MakeExclusive(Transitions − Com_N1,N2,nothing);}}
  36.else {
  37.∀P_i ∈ AfterPalces(a,N2) ; 1 ≤ i ≤ |AfterPalces(a,N2)|{
    38.∀t_j ∈ AfterTransitions(P_i,N2) ; 1 ≤ j ≤ | AfterTransitions(P_i,N2)|{
    39.if(j = 1)
      40.if(t_j ∈ Dif_N1){
        41.if(flag(t_j) ≠ 1){
          42.AddTransition(t_j,N1) ;
          43.p' = AddPlace(N1);
          44.AddArc(p', t_j,N1);
          45.AddArc(a,p',N1);}
        46.else {
          47. p' = BeforePlaces(t_j,N1);
          48.if( p' ∉ AfterPalces(a,N1))
          49.AddArc(a,p',N1)}; }
      50.else if(AfterPalces(t_j,N1) > 1)
        51.p' = SetAND(N1,null);//in this situation return End Place of And
      52.else
        53.p' = AfterPalces(t_j,N1);
      54.AddArc(a,p',N1);}
    55.if(j > 1)
      56.if(t_j ∈ Dif_N1){
        57.if(flag(t_j) ≠ 1){
          58.AddTransition(t_j,N1);
          59.AddArc(p',t_j,N1);}
        60.else
          61.Merge(BeforePalces(t_j),p')}
      62.if(AfterPalces(t_i,N1) > 1)
      63.SetAND(N1,p')}}}//in this situation return End Place of And
64.flagR(a) = 1}
```

Fig. 15- the ConsiderDif_{N_2} algorithm pseudo code

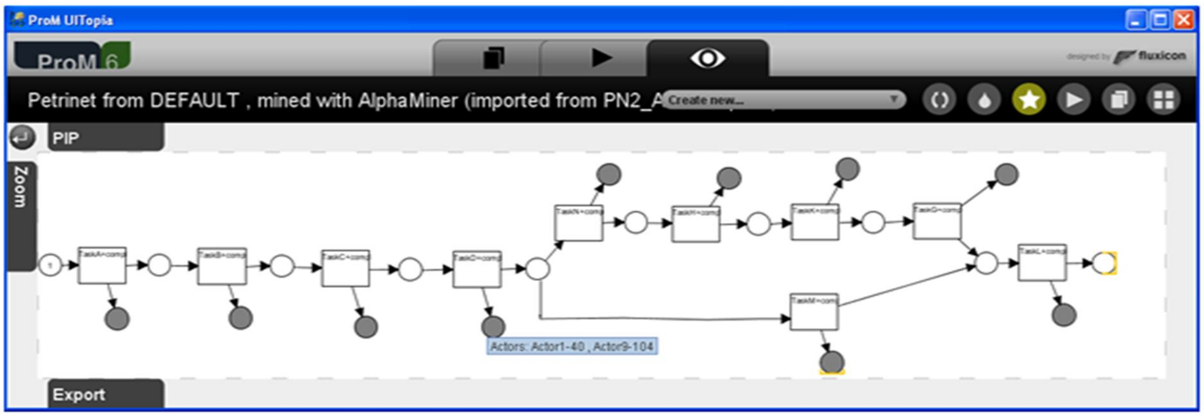

ProM UITopia
ProM 6
Petrinet from DEFAULT , mined with AlphaMiner (imported from PN2_A
Create new...
PIP
Zoom
Export
Actors: Actor1-40 , Actor9-104

Fig. 18- multi perspective model 2

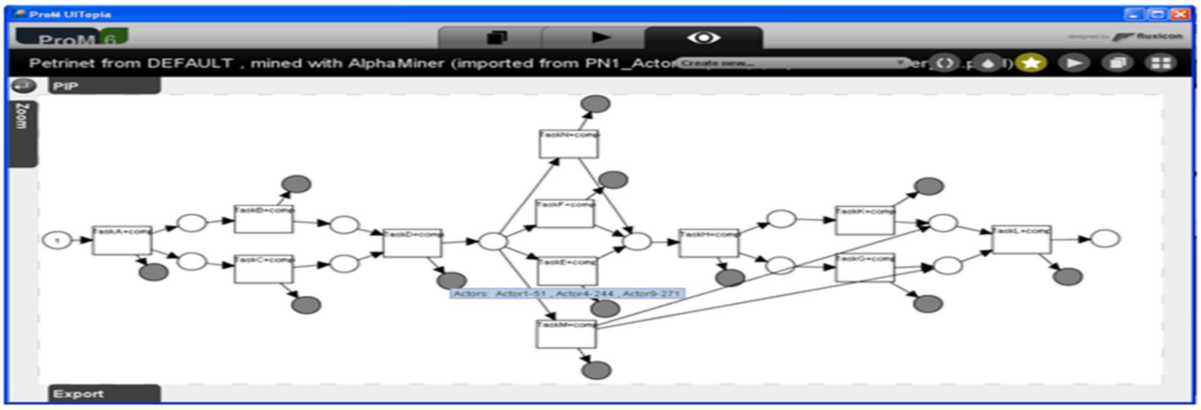

ProM UITopia
ProM 6
Petrinet from DEFAULT , mined with AlphaMiner (imported from PN1_Actor
PIP
Zoom
Export
Actors: Actor1-51 , Actor4-244 , Actor9-271

Fig. 19- the multi perspective model 1-2 (the result of comparing and merging models 1 and 2)

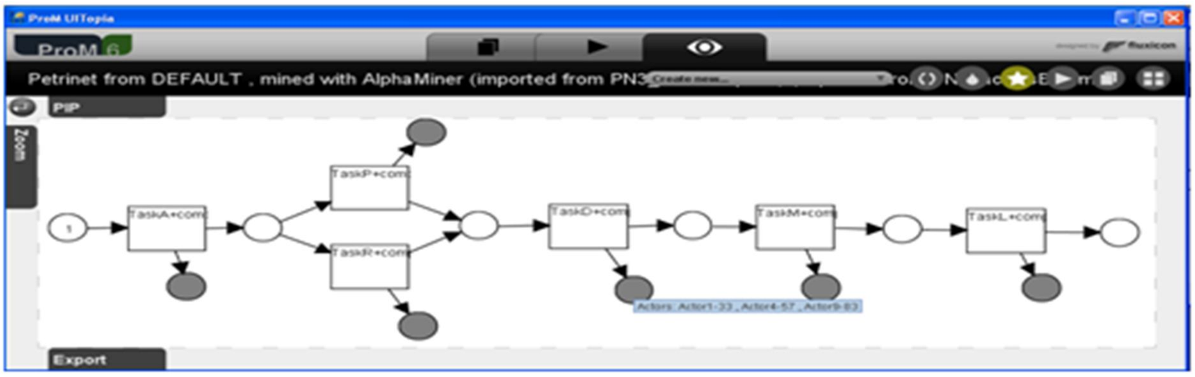

ProM UITopia
ProM 6
Petrinet from DEFAULT , mined with AlphaMiner (imported from PN3
PIP
Zoom
Export
TaskA+com
TaskP+com
TaskR+com
TaskD+com
TaskM+com
TaskL+com
Actors: Actor1-33 , Actor4-57 , Actor9-83

Fig. 20- multi perspective model 3

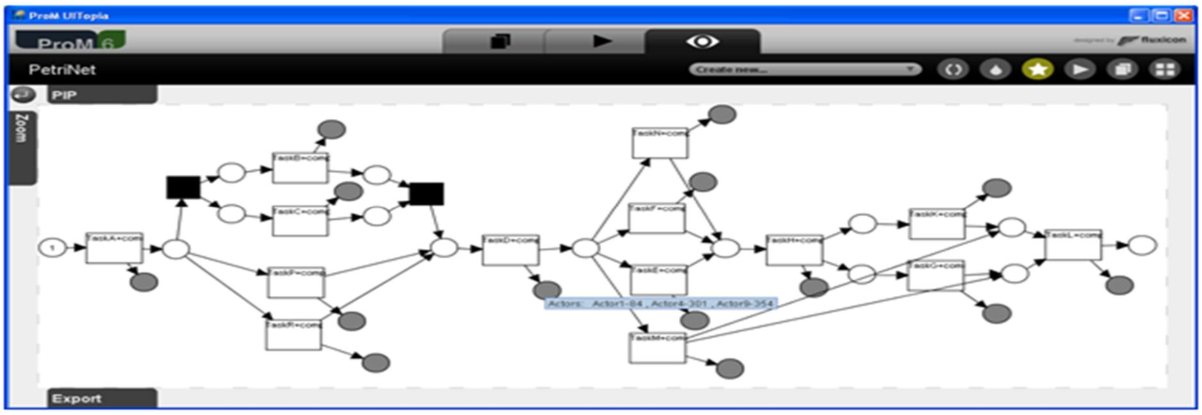


Fig. 21- multi perspective model 1-2-3 (the result of comparing and merging multi perspective models 1-2 and 3) In this section the operation and outputs (2 merged model) of the proposed algorithm are considered