Developing Workflow Patterns Based on Functional Subnets and Control – Flow Patterns

Riyanarto Sarno, Widyasari Ayu Wibowo, Dwi Sunaryono, Abdul Munif
Informatics Department, Faculty of Information Technology, Institut Teknologi Sepuluh Nopember, Indonesia
riyanarto@if.its.ac.id, widyasari.wibowo11@mhs.if.its.ac.id, dwi@if.its.ac.id, munif@if.its.ac.id

***Abstract*— Nowadays business processes are main concern in enterprises systems. The growing systems affect business process changes. Business processes always grow in order to fulfill the business activities. In this paper, we use a graph model called Petri Nets to analyze complex business processes. A Petri Net is the most usable model which often represents a process system in business concerns or others. Business processes are modeled using Petri Nets; then, the Petri Nets are separated into simpler fragments in order to make the analysis easier. In this context, we propose a method for developing an effective and efficient workflow repository system by using minimal functional subnet concept for generating common workflow in Petri Nets models. Thus process could lead to compose a new workflow. The results show that a complex business process can be decomposed into functional subnets which are smaller as well as simpler fragments of a complex business process. Then, the functional subnets are classified based on control-flow patterns in order to identify subnet roles in work-flow management. Later, we extract common fragments of functional subnets and use them for composing a new models of Petri Nets workflow required by user.**

Keywords—Petri Nets; Decomposition; Functional Subnet; Control-flow Pattern

I. INTRODUCTION

In this era, we often face difficulties to construct business process models [1,2] representing daily workflow occurred in companies. We need to be able to decompose the real daily workflows into valuable fragments to construct information needed. The decomposition of workflows can also be useful in process mining [3]. Petri Nets model presentation of business workflow using graphical and mathematical notation to give detailed information related to process system [4,5].

Petri Net often use to describe complex process modelling. Petri Nets is applicable for business process modelling such as modelling ERP system [6, 7]. In [8] there are three different kinds of metric: (i) label matching similarity which contrast the activity name attached to process activity; (ii) structural similarity which contrast activity labels as well as the arrangement of process models position in a workflow; and (iii) behavioral similarity which contrast element labels as well as causal interactions captured in the process model. The other possible approach of clustering similarity of business process is using the hierarchical merged models from the process clusters [9].

Besides the clustering based on similarity, there is another way for breaking down complexity of a large business process modeled by Petri Nets. Convenient approaches for decomposing using Petri Nets were carried out by several researchers. Van der Aalst [10] proposed a process mining method involving Petri Net decomposition. Process mining usually implements Petri nets as its analyzing model. Various algorithms have been enacted Petri nets as the internal representation used for process mining e.g the region-based process discovery techniques, α algorithm, and various conformance checking techniques. In [11, 12] Zaitsev presented the method for calculating Petri Nets invariants through decomposition-based technique. The result was Petri Nets could be composed from invariants of its functional subnets.

Functional subnet in [11,12] will mean nothing in work-flow process unless we find the relation between functional subnet and control-flow pattern. The pattern is used as specific expression when executing a process model. There are 20 control-flow patterns available for evaluation of work-flow management system [13]. Each of them is represented with a graphical notation called WPSL, *Workflow Pattern Specification Language,* which is stately explained and defined with visual annotation in graphic which is used as manner for analyzing and assessing the differences in the modelling languages.

This paper is intended to provide some illustrations about how the decomposition works and how it can help to solve another problem in the process-related problem. The first part of this paper will discuss the introduction about this research. Then we discuss about several related studies of this research. The next part will discuss about the method we use in decomposition of Petri Nets including the relation with control flow patterns. In evaluation we provide an experiment implementing the proposed method. Finally, in last section, we will present the conclusion of this research as well as some future research plans.

II. RELATED WORK

In [12], Zaitsev introduced the concept of Petri Net decomposition, in which any petri net can be factored into its functional subnet. The resulting decomposition may give a nice partition of the edges that enables us to understand a whole giant and complex process by dividing it into some relatively small subnet. The decomposition may also help running the process mining algorithm into several parts that is done in parallel.

978-1-4799-8386-5/15/$31.00 ©2015 IEEE

To describe the method rigorously, we first define several notions and definitions.

A. Definition and Notation of Petri Net

On the formula, the *Petri Nets* is defined as a triplet N = (P, T, F) where $F \subseteq P \times T \cup T \times P$ .The composite of *places* is annotated as P, whereas the set T annotates *transitions* set. F represents the set of relation which connect a place to a transition or vice versa. For convenience, for every $u \in P \cup T$, we define by (1) and (2).

$$\bullet u = \{v \in P \cup T \mid (v, u) \in F\} \quad (1)$$
$$u \bullet = \{v \in P \cup T \mid (u, v) \in F\} \quad (2)$$

The set $\bullet u$ and $u\bullet$ is said to be the *input* set and the *output* set of u respectively.

A *functional net* is defined by the triplet Z = (N, X, Y) whereas Petri Nets represents with, N = (P, T, F), $X, Y \subseteq P$ and $X \cap Y = \emptyset$ such as in (3).

$$Y\bullet = \emptyset \text{ and } \bullet X = \emptyset \quad (3)$$

X is said to be the *starting place set*, Y is said to be the *ending place set*. Additionally we also define $Q = P - (X \cup Y)$ as the *internal place set*. Moreover, naturally we also define $X \cup Y$ as the *external place set*.

N' = (P', T', F') is called a subnet of N = (P, T, F) if $P' \subseteq P$, $T' \subseteq T$, $F' \subseteq F$. A subnet *induced* by $T' \subseteq T$ is the subnet which contain T and every places and transitions in N that is adjacent to every place in T.

A functional subnet of Z = (N', X, Y) is called the *functional subnet* of a petri net N, denoted by $Z \prec N$ if Z is a subnet of N and every vertex (place or transition) in Z is connected to every other vertex in N - Z only by the vertices in X or Y. A functional subnet $Z' \prec N$ is *minimum* if Z' does not contain any other functional subnet in N.

B. Some Fundamental Properties of Minimal Functional Subnet

Since the purpose of this article is to study the decomposition of petri net using the concept of minimal functional subnet, we first describe some elements in the petri net that is being decomposed by the minimal functional subnet. In [11], Zaitsev gave a fundamental proposition to state subnet functional forming.

III. THE PROPOSED METHOD

A. State of The Art

In developing workflow pattern we have to be aware of reusability aspect in every model variation. A complex model of business process is composed from several simpler fragments. Many variations could be composed from common simpler fragments with other variant fragments addition. This concept might be simplify the way of analyzing a complex business process model. We conclude theory above in process flowchart in Fig. 1.

There are two phase in creating reusable fragments for business process model variations. In the first step we decompose a complex model into simplbodyer form called Functional Subnet and find the commonly used fragments. Those common Functional Subnet then taken into second phase for mapping with Control Flow Pattern resulting common Control Flow Subnets which can be used for composing other models variation.

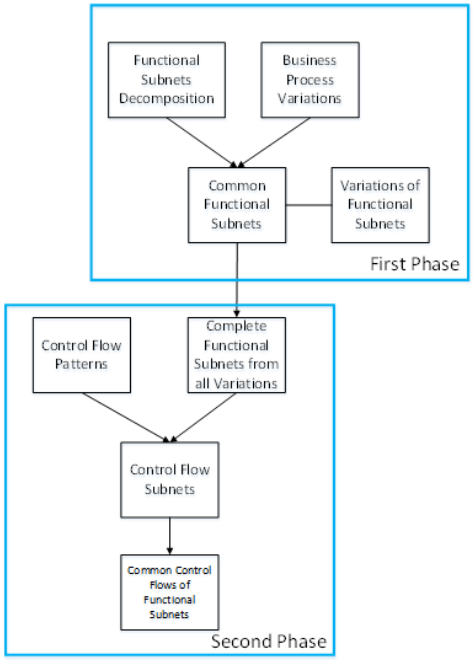

Fig. 1. Process to create reusable fragments for business process model variations

B. The Algorithm of Functional Subnet Decomposition

In [11], Zaitsev introduced a polynomial algorithm to compute every minimal functional subnet of any arbitrary petri net. Since the algorithm decompose Petri Nets into simplify form, the original algorithm which Zaitsev proposed couldn't show arbitrary pattern in control flow pattern (shown in Fig. 9). Thus we added a modification step so the algorithm could produce arbitrary cycle pattern.

ALGORITHM 1.

Input : Petri Nets N = (P, T, F) and $t \in T$

Output : Minimal functional subnet of N which contains t, Z = (X, Q, Y, R), where $X \subseteq P$ and $X = \bullet R$, $Y \subseteq P$ and $Y = R\bullet$, $Q \subseteq X, Y$, $R \subseteq T$, $\Sigma = t \in \bullet X$, and $\beta = t \in Y\bullet$

Step 0. Select a random transition $t \in T$ from Petri Nets N (choose the first transition to the last one in the net sequentially) and add up transition to the set of selected transitions R: = {t}.

Step 1. Build the functional subnet Z yielded by the set R: Z = B (R) = (X, Q, Y, R).

Step 2. If Z is settled in N, therefore Z is the only component of functional subnet. Do check if $\beta \subseteq \Sigma$ and $R \neq \beta$, subnet Z is part of arbitrary cycle pattern. Thus subnet Z will be merged with other subnet creating arbitrary cycle such that Z: = $Z_R \cup Z_\Sigma$. Else presume to step 3.

Step 3. Construct the set of assimilated transitions

$$S = \{t \mid t \in X\bullet \wedge t \quad R \vee t \in \bullet Y \wedge t \quad R \vee t \in \bullet Q\bullet \wedge t \quad R\}$$

Step 4. Pressume that R: = R ∪ S and repeat from step 1

C. Control-flow Patterns

Based on [13] there are 20 patterns that could be used to identify work-flow process on a process model. Unfortunately not all 20 patterns available is practical for Petri Nets approach. Thus we made a selection of patterns which could be apply using Petri Nets (see Fig.2 to Fig. 9).

1) CF1 – Sequence

All tasks in an activity workflow pattern is done sequentially. Current activity is executed after preceding activity is complete in the identical process.

Fig. 2. Sequence pattern in Petri Nets model

2) CF2 – Parallel Split

Parallel split is a branching point inside activity workflow pattern where a solitary flow of control divides into numerous flows. Those flows run together in parallel, therefore allow all tasks to be obtained in any order.

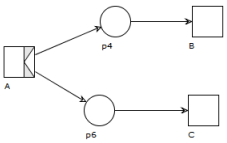

Fig. 3. Parallel split pattern in Petri Nets model

3) CF3 – Synchronization

A conjoint point inside the activity workflow pattern where numerous collateral flows or task concatenate into single flow of control, therefore synchronizing various flows.

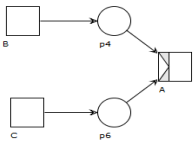

Fig. 4. Synchronization pattern in Petri Nets model

4) CF4 – Exclusive Choice

A division spot inside activity workflow pattern where the flow splits into several options drive by a logical judgment or workflow control data. This control flow only executes one of several available options.

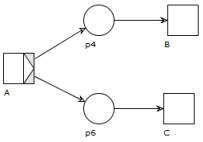

Fig. 5. Synchronization pattern in Petri Nets model

5) CF5 – Simple Merge

A conjoint spot inside workflow process where two or more auxiliary flows executed in conjunction without synchronization. In this pattern only one activity is executed.

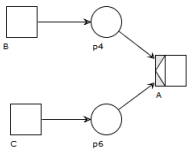

Fig. 6. Simple merge pattern in Petri Nets model

6) CF6 – Multiple Choice

A branching spot inside activity workflow pattern, where the flow splits into several options drive by a logical judgment or workflow control data. One or more auxiliary flow are executed together.

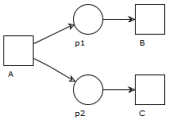

Fig. 7. Multiple choice pattern in Petri Nets model

7) CF7 – Synchronizing Merge

A splitting spot inside workflow activities where various flows join into single flow. When at least two activity is executed, synchronization of the active flows is necessarily taken. On the contrary, when only one activity is done, the flow will re-converge without synchronization.

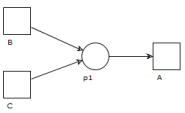

Fig. 8. Synchronizing merge pattern in Petri Nets model

8) CF8 – Arbitrary Cycles

A repetitive point inside workflow activities resulting some activities can be done recursively.

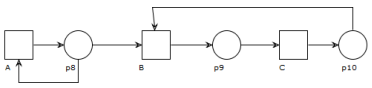

Fig. 9. Arbitrary cycles pattern in Petri Nets model

IV. Evaluation

A. Decomposition of Petri Nets

In this experiment we consider 5 examples (see Fig. 10 to Fig. 14) of different process models.

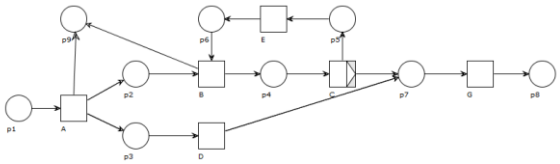

Fig. 10. Process model Variation 1

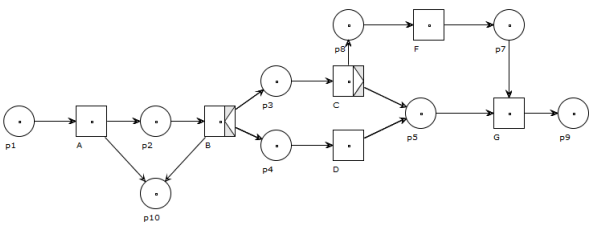

Fig. 11. Process model Variation 2

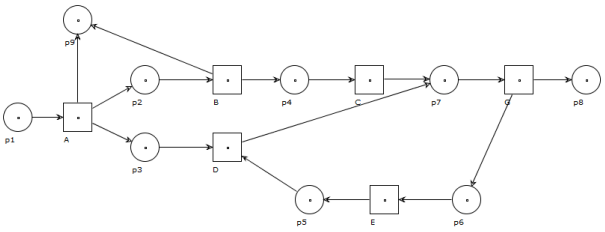

Fig. 12. Process model Variation 3

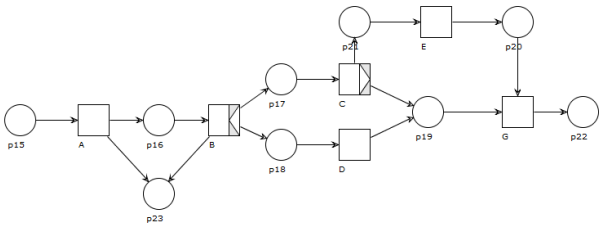

Fig. 13. Process model Variation 4

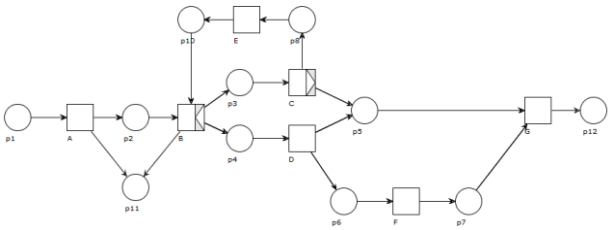

Fig. 14. Process model Variation 5

For every process model variation given above we decompose the nets into functional subnet based on **Algorithm 1**. We give step by step explanation to decompose Fig. 10 below:

1. Define $R_1 = \{A\}$.
2. Subnet $Z_1 = (X_1, Q_1, Y_1, R_1)$ is built by R_1, with the result that $X_1 = \{p_1\}$, $Q_1 = \{\}$, $Y_1 = \{p_2, p_3, p_9\}$, $R_1 = \{A\}$.
3. Z1 is not complete since $B \in \bullet Y$ and $B \notin R_1$
4. $S = \{B\}$
5. $R_1 := R_1 \cup S$
6. Subnet $Z_1 = (X_1, Q_1, Y_1, R_1)$ is built by R_1, so $X_1 = \{p_2\}$, $Q_3 = \{p_1, p_3\}$, $Y_3 = \{p_9, p_4\}$, $R_1 = \{A, B\}$. Subnet Z_1 is shown in Fig. 15.

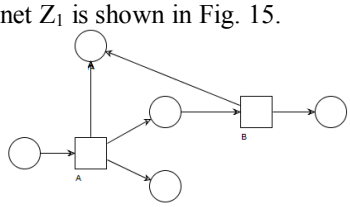

Fig. 15. Functional subnet Z_1 for model Variation 1

7. Define $R_2 = \{C\}$
8. Subnet $Z_2 = (X_2, Q_2, Y_2, R_2)$ is built by R_2, so $X_2 = \{p_4\}$, $Q_2 = \{\}$, $Y_2 = \{p_7, p5\}$, $R_2 = \{C\}$.
9. Z_2 is not complete, since $D \in \bullet Y$ and $D \notin R_2$
10. $S = \{D\}$
11. $R_2 := R_2 \cup S$
12. Subnet $Z_2 = (X_2, Q_2, Y_2, R_2)$ is built by R_2, so $X_2 = \{p_3\}$, $Q_2 = \{p_4, p_5\}$, $Y_2 = \{p_7\}$, $R_2 = \{C, D\}$. Subnet Z_2 is shown in Fig. 16.

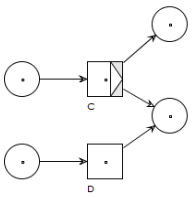

Fig. 16. Functional subnet Z_2 for model Variation 1

13. Define $R_3 = \{E\}$
14. Subnet $Z_3 = (X_3, Q_3, Y_3, R_3)$ is built by R_3, so $X_3 = \{p_5\}$, $Q_3 = \{\}$, $Y_3 = \{p_6\}$, $R_3 = \{E\}$. Subnet Z_3 is shown in Fig. 17.
15. Z_3 is complete

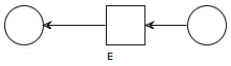

Fig. 17. Functional subnet Z_3 for model Variation 1

16. Define $R_4 = \{G\}$
17. Subnet $Z_4 = (X_4, Q_4, Y_4, R_4)$ is built by R_4, so $X_4 = \{p_7\}$, $Q_4 = \{\}$, $Y_4 = \{p_8\}$, $R_4 = \{G\}$. Subnet Z_4 is shown in Fig. 18.
18. Z_4 is complete

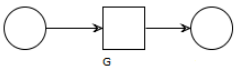

Fig. 18. Functional subnet Z_4 for model Variation 1

Then with the same **Algorithm 1** we obtain functional subnets for model variation 2, 3, 4, and also 5.

B. Common Functional Subnets

Before deciding common functional subnets in each variants, we need to compute similarity value for each variants. Therefore we calculate similarity value based on structural aspect as in (4).

$$sim = \frac{same\ parts\ of\ nets}{same\ parts\ of\ nets + different\ parts\ of\ nets} \quad (4)$$

TABLE I. SIMILARITY VALUE BETWEEN VARIATIONS

Variation / Variation	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5
Variant 1	1	0.42	0.78	0.46	0.39
Variant 2		1	0.39	0.82	0.61
Variant 3			1	0.42	0.37
Variant 4				1	0.61
Variant 5					1

Based on Table I we decide on a parameter value to form a group consist of several most similar model variations. We decide 0.60 as parameter value, such that two model variations which have similarity value above 0.60 could be group in one cluster. Therefore we decide on two cluster: **Cluster 1** consists of model Variation 1 and Variation 3; **Cluster 2** consists of model Variation 2, Variation 4, and Variation 5.

From Cluster 1 we could retrieve common functional subnets from Variation 1 and Variation 3 which are shown in Fig. 19.

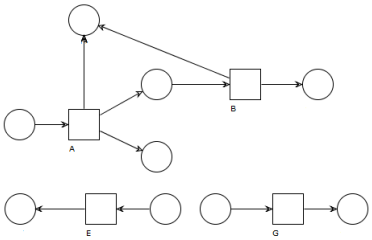

Fig. 19. Common functional subnets for Cluster 1

As for Cluster 2 we also retrieve common functional subnets from Variation 2, Variation 4, and Variation 5 which are in Fig. 20:

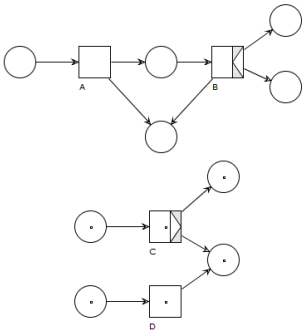

Fig. 20. Common functional subnets for Cluster 2

C. *Control-flow Patterns Mapping*

Since the common functional subnets for each cluster are already known, we map them to control-flow patterns which already decided earlier above (shown in Fig. 2 to Fig. 9). The result of mapping is called Control Flow Subnet. Each Control Flow Subnet only represent one Control Flow Pattern. For every process model Variations in Fig. 10 to Fig. 14, we map them with Control Flow Patterns, resulting Control Flow Subnets as in Fig. 21 to Fig. 25.

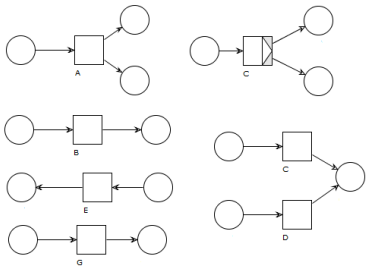

Fig. 21. Control Flow Subnet for Variation 1

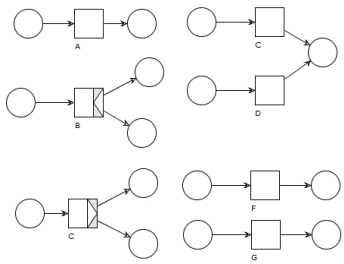

Fig. 22. Control Flow Subnet for Variation 2

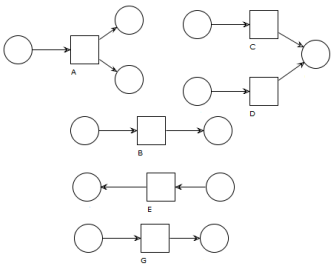

Fig. 23. Control Flow Subnet for Variation 3

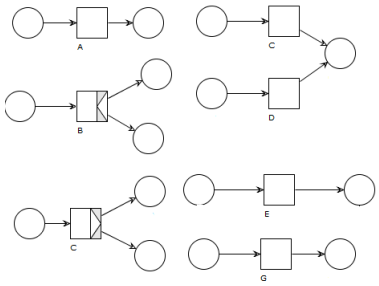

Fig. 24. Control Flow Subnet for Variation 4

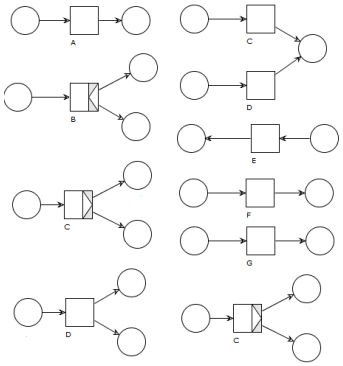

Fig. 25. Control Flow Subnet for Variation 5

By eliminating Control Flow Subnet which has the same activity name label we could simplify model composition as shown in Fig. 26.

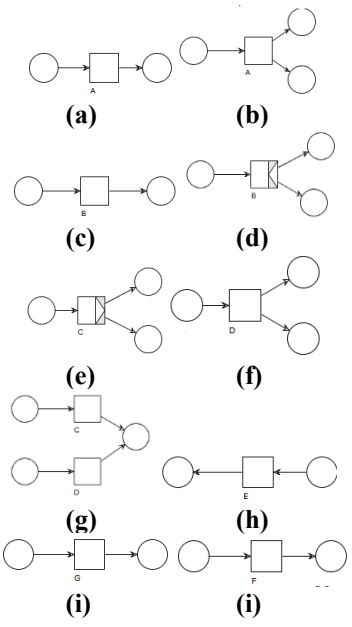

Fig. 26. Control Flow Subnets represent in Compress

D. Common Control Flow Subnet

After finding all Control Flow Subnet in every model variations, we could extract the most commonly used Control Flow Subnet in 5 Variations. Based on Fig. 26 (a) to Fig. 34 (j) we made a distribution list (show in Table II) to show how often a type of Control Flow Subnet is used in a model variation.

TABLE II. DISTRIBUTION OF CONTROL FLOW SUBNET

Control Flow Patterns / Variation	a	b	c	d	e	f	g	h	i	j
Variation 1	0	1	1	0	1	0	1	1	1	0
Variation 2	1	0	0	1	1	0	1	0	1	1
Variation 3	0	1	1	0	0	0	1	1	1	0
Variation 4	1	0	0	1	1	0	1	1	1	0
Variation 5	1	0	0	1	1	1	1	1	1	1
Total	3	2	2	3	4	1	5	4	5	2

From Table II, we decided a common Control Flow Subnet for Variation 1 to Variation 5 is a Control Flow Subnet which is used at least in more than 50% of number of Variations. Since we have 5 Variations, Control Flow Subnet is called common whenever it appeared in at least 3 from 5 Variations. Hence we have type a, d, e, g, h, and i (see Fig. 27) as common Control Flow Subnet.

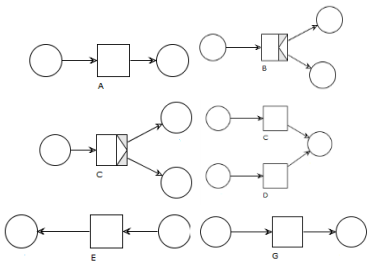

Fig. 27. Common Control Flow Subnets from figure 27(a), 27(d), 27(e), 27(g), 27(h), 27(i)

Those common Control Flow Subnets could be composed in a single model (see Fig. 28) called common process as shown below.

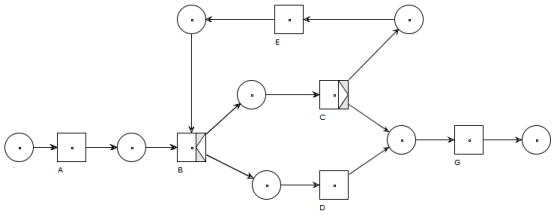

Fig. 28. Common process derived from variations

V. CONCLUSION

The paper presented a method to decompose complex business process into smaller fragments which are called functional subnets. A business process was modeled using Petri Nets; hence, the Petri Net was decomposed into functional subnets using certain polynomial complexity algorithm. The functional subnets from several business process variations were extracted to obtain the common subnets. Afterwards, the common subnets were mapped into Control Flow Patterns, and resulting new fragments called Control Flow Subnets. Therefore, the common Control Flow Subnets have significant reusability for composing new models. Thus, a new workflow model can be composed by adding common Control Flow Subnets and several different variations. This approach could develop efficiently and effectively a new workflow.

ACKNOWLEDGEMENT

We would like to thank Institut Teknologi Sepuluh Nopember and DIKTI - Directorate General of Higher Education - Ministry of Education and Culture of Indonesia for supporting the research.

REFERENCES

[1] van der Aalst W.M.P., van Dongen B.F., Herbst J., Maruster L., Schimm G., Weijters A.J.M.M., Workflow Mining : A Survey of Issues and Approaches, Data aand Knowledge Engineering, Volume 47, No. 2, 2003, pp. 237-267.

[2] Murata Tadao, Petri Nets : Properties, Analysis and Application, Proceedings of The IEEE, Vol. 77, No. 4, 1989, pp. 541-580.

[3] Sarno Riyanarto, Djeni C.A., Mukhlash,I., Sunaryono D., "Developing A Workflow Management System for Enterprise Resource Planning," *Journal of Theoretical and Applied Information Technology,* February 2015, Vol. 72, No.3, pp. 412-421

[4] Sarno Riyanarto, Dewandono D., T. Ahmad M. N., & Sinaga F., Hybrid Association Rule Learning and Process Mining for Fraud Detection, *IAENG* International Journal of Computer Science, Vol 42, No. 2, 2015, pp 59-72

[5] Bobbio Andrea, System Modelling with Petri Nets, System Reliability Assessment, Kluwer p.c., 1990, pp. 102-143,.

[6] van der Aalst W.M.P., Decomposing Petri Nets for Process Mining : A generic Approach, Technical Report – BPM Center, 2012, pp. 1-3.

[7] Sarno Riyanarto, Sanjoyo Bandung Arry, Mukhlash Imam, Astuti Hanim Maria, Petri Net Model of ERP Business Process Variation for Small and Medium Enterprises, Journal of Theoritical and Applied Information Technology, Vol. 54, No. 1, 2013, pp. 31 – 38.

[8] Sarno Riyanarto, Ginardi Hari, Pamungkas Endang Wahyu, Sunaryono Dwi, Clustering of ERP Business Process Fragments, *Proceeding IEEE International conference on computer, control, informatics, and its applications*, 2013, pp. 319-324.

[9] Jung J.Y., Bae J., and Liu L., Hierarchical Clustering of Business Process Models, SCC 2008, IEEE International Conference on Volume 2, 2008, pp. 613 – 616.

[10] Dijkman R., Dumas M., van Dongen B., Kaarik R., and Mendling J., Similarity of Business Process Model : Metric and Evaluation, Information Systems 36(2), 2011, pp. 498-516.

[11] D. A. Zaitsev, Decomposition-based Calculation of Petri Net Invariants, Proceedings of Workshop on Token based computing of the 25-th International conference on application and theory of Petri nets, Bologna, Italy, 2004, pp. 79-83.

[12] D. A. Zaitsev, Decomposition of Petri Net. Cybernetics and Systems Analysis, Vol. 40, No. 5, 2004, pp. 739-746.

[13] Mulyar Nataliya, van der Aalst W.M.P., H.M. ter Hofstede Arthur, Russel Nick, Towards a WPSL : A Critical Analysis of the 20 Classical Workflow Control-flow Pattern, 2006, pp.1-66.