Discovering Redo-Activities and Performers' Involvements from XES-Formatted Workflow Process Enactment Event Logs

Dinh-Lam Pham[1], Hyun Ahn[1], and Kwanghoon Pio Kim[1*]
[1]Collaboration Technology Research Lab,
Division of Computer Science and Engineering, KYONGGI UNIVERSITY,
154-42 Youngtong-Gu Suwon-Si Gyeonggi-Do, 16627, Republic of Korea
[e-mail: {phamdinhlam, hahn, kwang}@kgu.ac.kr]
*Corresponding Author: Kwanghoon Pio Kim

Received February 27, 2019; revised April 18, 2019; accepted June 6, 2019; published August 31, 2019

Abstract

Workflow process mining is becoming a more and more valuable activity in workflow-supported enterprises, and through which it is possible to achieve the high levels of qualitative business goals in terms of improving the effectiveness and efficiency of the workflow-supported information systems, increasing their operational performances, reducing their completion times with minimizing redundancy times, and saving their managerial costs. One of the critical challenges in the workflow process mining activity is to devise a reasonable approach to discover and recognize the bottleneck points of workflow process models from their enactment event histories. We have intuitively realized the fact that the iterative process pattern of redo-activities ought to have the high possibility of becoming a bottleneck point of a workflow process model. Hence, we, in this paper, propose an algorithmic approach and its implementation to discover the redo-activities and their performers' involvements patterns from workflow process enactment event logs. Additionally, we carry out a series of experimental analyses by applying the implemented algorithm to four datasets of workflow process enactment event logs released from the BPI Challenges. Finally, those discovered redo-activities and their performers' involvements patterns are visualized in a graphical form of information control nets as well as a tabular form of the involvement percentages, respectively.

***Keywords*:** Process mining, redo-activities, bottleneck analysis, information control net, XES

A preliminary version of this paper was presented at ICONI 2018 and was selected as an outstanding paper. This version includes detailed analysis, comparison of redo-activities between four dataset and results about performers' involvement in redo-activities. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (Grant No.2017R1A2B2010697). This work also was supported by Kyonggi University's Graduate Research Assistantship 2019.

http://doi.org/10.3837/tiis.2019.08.016 **ISSN : 1976-7277**

1. Introduction

In the last few years, workflow process mining lets us know and more understand our business system from the history logs. From a very complicated relationship in the workflow log, with process mining techniques, we apprehend more information about what is happening in the business system. As a consequence, we can analyze, make visualization for the whole system, compare it to the predefined workflow model. Process mining also helps us recognize which part of the system shows good performance or is needed to be improved by comparing the performance (e.g., time and cost) between the planned and observed behaviors. Thus, we can make appropriate adjustments by using the analyzed results from process mining activities. One of the problems to be solved in the workflow process mining field is to identify the bottleneck points of the system which make the system slowly, increase system cost, and effect to the other components [1][2]. A bottleneck point may be in the form of paper documents, the transfer of work among activities affecting subsequent jobs. Bottleneck point may also occur because previous work has not been processed and the next job is required. It leads to a congested situation, affecting the entire system performance. The factor that often causes a bottleneck point in a workflow process model is redo-activities which need to be performed again even though it has already been done in the previous step. The reasons are the work has not finished on time, the quality is not as expected, missing some documents, and some mistakes possibly. Redo-activities usually make increasing cost, working slowly and effect to the later process.

In this paper, we propose an approach to identify redo-activities from the workflow process logs, representing these objects under the information control net model. Also, we present a couple of algorithms and experimental analyses to identify the involvements of these redo-activities. We try to analyze and find out the correspondence between performers, execution time and the repetition of activities, thereby helping to find out potential activities in the bottleneck points of the system.

The rest of the paper is organized as the following. Next section reviews background, related work and motivation. Section 3 proposes an approach to discover the redo-activities and shows experimental results with some datasets. Section 4 analyzes the performers' involvements in redo-activities. Finally, Section 5 summarizes the paper.

2. Background, Related Work and Motivation

2.1. Background

The theoretical background is the information control net (ICN) methodology [3]. ICN is a typical workflow modeling approach supporting graphical and formal representations. **Fig. 1** illustrates the ICN primitives. In this model, there are four types of control-flow:

- Sequential control-flow: An activity follows another activity by a sequential order.
- OR control-flow: An activity follows a disjunctive (or decision) transition followed by two or more activities. Only one activity of the followed activities will be executed at the next step.
- AND control-flow: An activity follows a conjunctive (or parallel) transition followed by two or more activities. All the followed activities will be executed at the next step in parallel.
- Loop control-flow: An activity follows a loop transition. The activities included the loop block will be executed repeatedly until the termination condition holds true.

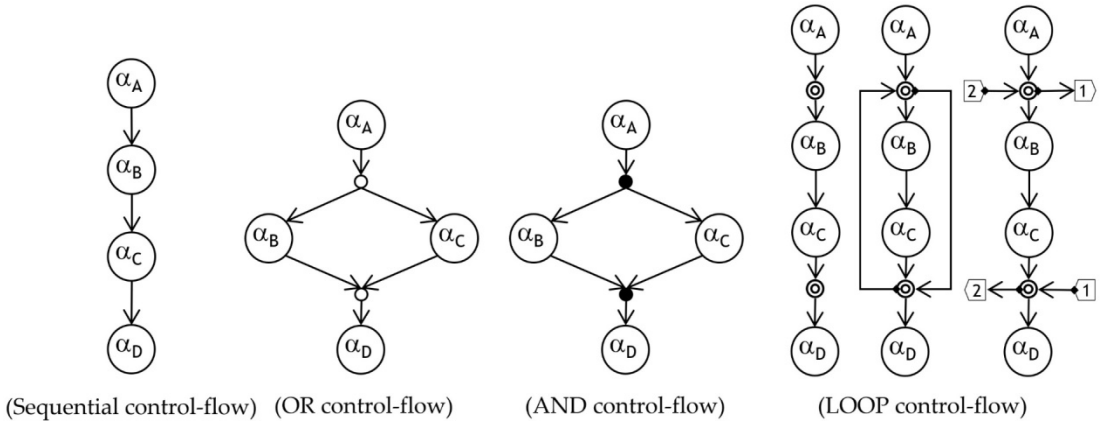


Fig. 1. Control-flow primitives of the ICN model [3].

As shown in **Fig. 2**, we present an example of a simple paper review process represented as an ICN model. As details of the process, the author first submits the paper (A), then the peer review is started (P). At the next step, a disjunctive transition (OR control-flow) occurs according to the peer review result, divided into "*paper accepted*", "*paper accepted with revision*", and "*paper rejected*".

- *Paper accepted*. The reviewers accept the paper without any revisions (G). Therefore, it is the best result for authors. The only one step remaining is the submission of the camera-ready version of the paper (H). Finally, the submission process will be terminated (Z).
- *Paper accepted with revision*. The reviewers accept the paper, but they require some revisions (D). Accordingly, the authors will revise their paper according to the requirements from the reviewers (E) and resubmit it (F). The reviewers will reexamine the modifications to their paper and make a decision whether the revision is acceptable or not. The revision and resubmit steps can be repeated (M) until the quality of the paper has evolved to an extent that does not require any modification. This repetitive behavior is represented by a Loop-open and Loop-close control-flow types. However, if the author's revised paper does not meet the requirements after a few times of revising or the editing is not on time, the paper finally might be rejected. We do not take into account this exceptional situation to the process example.
- *Paper rejected*. If the paper does not meet the requirements for the acceptance, the paper will be rejected (R) and the process will be terminated automatically (Z).

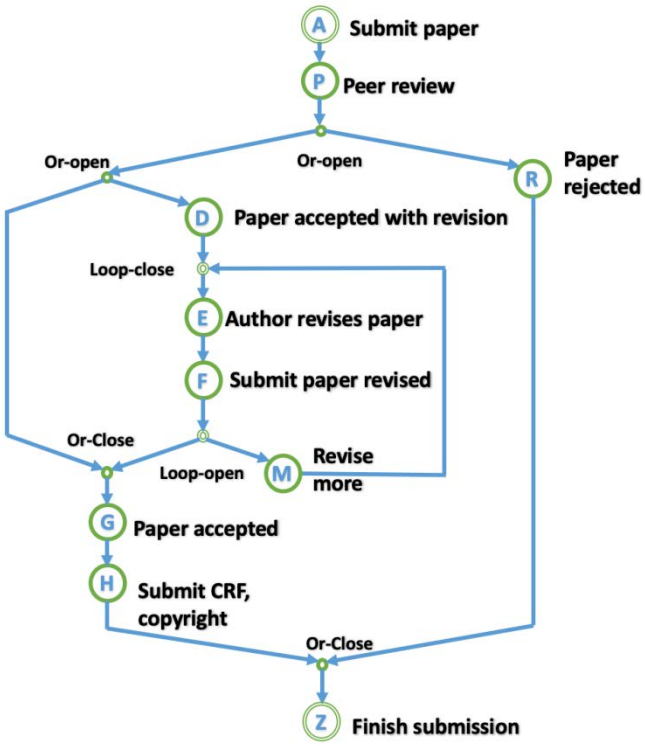


Fig. 2. ICN model of the simple paper review process.

2.2 Related Work and Motivation

In the field of process mining, there are several process mining techniques have been introduced to discover process model, reconstruct, noise filter, improve the performance of the information system from the event logs such as the alpha (α) algorithm [4], heuristic mining [5], fuzzy mining [6], genetic process mining [7]. Moreover, discovering the bottleneck area has an essential meaning in improving the information system performance. It helps us to find out some position which needs to modify, upgrade in the hole system. Several research groups try to deal with how to realize the bottleneck area, improve the effectiveness and efficiency:

Aalst [8] introduced the process mining spectrum with several techniques to identify and understand bottlenecks, inefficiencies, deviations, and risks. In this architecture, timestamps can be used to compute bottlenecks and extend the model with performance information. Conformance checking can be used for auditing and compliance. By replaying the event log on a process model, it is possible to quantify and visualize deviations to detect bottlenecks and build predictive models.

Gupta et al. [9] conducted a case-study on data extracted from the Bugzilla issue tracking system of the popular open-source Firefox browser project. They tried to study self-loops, bottlenecks, metrics, and others to compute the degree of conformance between the design time and the runtime process. They observed self-loop and back-forth transitions (undesirable and inefficient) to identify bottlenecks to help process owner for improvement.

Mukhlash [10] proposed an approach to analyzing the business process of production systems by using Coloured Petri Nets (CPN). CPN can be used to analyze the performance of business processes. They collect data from the event log and build a model of Petri Nets then transformed it into CPN. They analyze the bottleneck in the model. Finally, they modified the existing business process model to improve its performance.

Detecting bottlenecks to improve the effectiveness and efficiency of the system has been sought by many research groups and has achieved specific results. All approaches have its effects on improving the operational efficiency of the system. In order to offer a different approach in determining bottlenecks, improving system performance, we propose an approach for discovering the redo-activities and performers' involvements. We observed that it is necessary to consider and analyze the information related to redo-activities which repeats itself. The repetition may be due to many different factors, but it affects the whole operation of the system and contribute to the bottleneck effect in the system. Alternatively, one of the problems in building the ICN model from the enactment logs is how to dig up the control flow and classify it (i.e., OR-control, AND-control, LOOP-control). Discovering redo-activities contribute to building Loop control-flows in the ICN model of the information system from workflow enactment event logs. In the next section, we describe the approach and algorithm discovering redo-activities from workflow enactment event log and its experimental result.

3. Discovering Redo-Activities from Workflow Enactment Event logs

3.1 Discovering Redo-Activities

If we only examine the raw data from workflow event logs, it is a complicated task to identify redo-activities. Since all the records of event logs are sorted by those time orders, it is possible to uncover all precedence relationships between activities by analyzing the attributes of each record (e.g., timestamp, resource, and id). Our solution first discovers the workflow process which is represented as a weighted directed graph (WDG) that depicts interactions indicating work transferences between activities, and those weighted values. In the graph, vertices and arcs indicate activities, work transferences between them, respectively. An arc connected from the vertex A to the vertex B indicates that the activity A is followed by the activity B. The value on the arc represents how many times the work transferences from activity A to activity B was occurred. Based on the analysis of such WDG, we can figure out that the activity A is a redo-activity, when and only when the value of the arc AA (represented as a self-loop) is larger than 0. The fact that the value of the arc AA is equal to 0 (the arc AA does not exist) indicates that the activity A was not repeated by itself.

The following algorithm describes the WDG discovery from workflow enactment logs formatted in XES (eXtensible Event Streams). In the algorithm, we use an adjacency list to store the graph.

Algorithm: Redo-Activity Discovery

Input: Workflow enactment event logs formatted in XES (**ωϵ**).
Output: A weighted directed graph (WDG).
Begin

```
1:    Open the enactment log file.
2:    Initialize all member in adjacency list β to ∅
3:    For(∀trace[i] ∈ (ωϵ))
4:      For(∀event[j] ∈ trace[i])
5:        currentActivity = event[j].GetActivityName()
6:        If(currentActivity ∉ β)
7:            β.add(currentActivity)
8:        Else
9:            listOfVertex = β.add(currentActivity)
10:           nextActivity = event[j+1].GetActivityName()
11:         If(nextActivity ∉ listOfVertex)
12:             listOfVertex.add(nextActivity)
13:         Else
14:             listOfVertex.Update(nextActivity, +1)
15:         Endif
16:       Endif
17:     Endfor
18:   Endfor
19:   Close the enactment log file
20:   Graphstream G.source = β          // Transfer adjacency list to G
21:   G.display()
```

End

As a running example of the algorithm, **Fig. 3** shows a WDG discovered from the event logs of the BPI 2018 dataset [12]. This graph represents the third trace (trace-3) in the event logs, and it includes 14 activities. As shown in this figure, we can intuitively recognize that there are three redo-activities (i.e., *mail valid*, *calculate* and *finish editing*). The detected redo-activites have been executed according to their occurences of two (*mail valid* and *calculate*) and three (*finish editing*), and we can interpret that there were repeated executions that cause the delays of total execution time of the corresponding trace.

To get a better understanding of the redo-activities that appear in the event logs, we apply the algorithm on four datasets provided by 4TU [12]. **Table 1** summarizes the brief information of the datasets used in this paper.

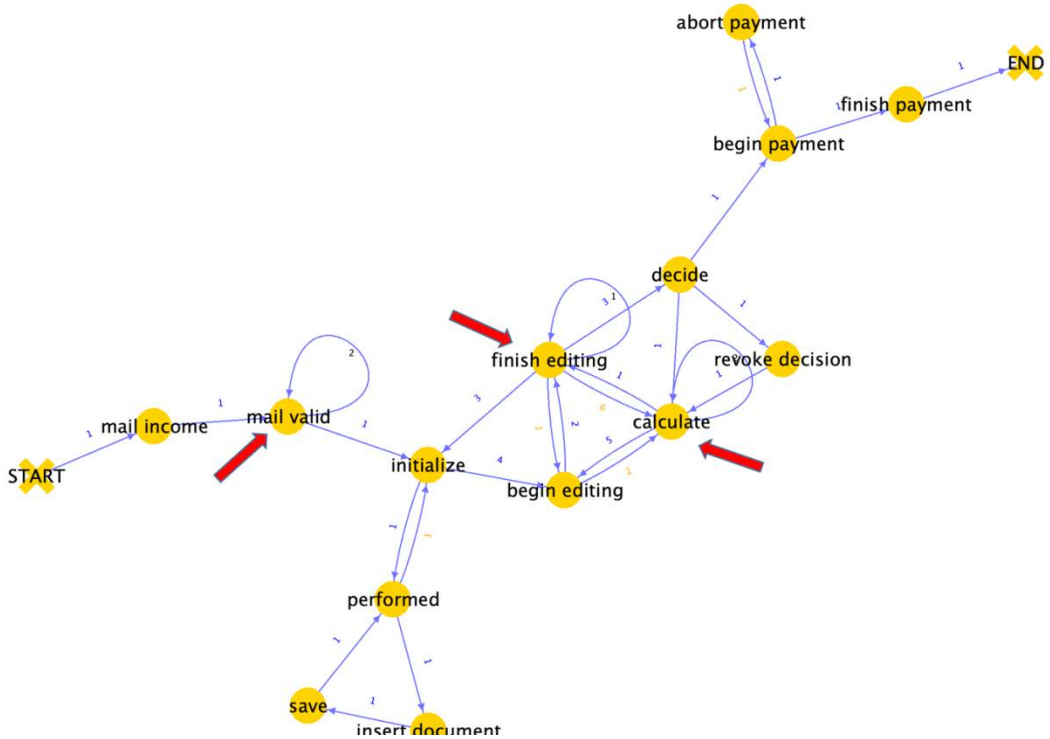


Fig. 3. Visualization of the WDG of the third trace (trace-3) in the BPI 2018 dataset [12].

Table 1. Summary of the event log datasets.

Log data	Num. of traces	Num. of events	Num. of activities	Num. of performers
BPI 2012	13,087	262,200	24	68
BPI 2017	31,509	1,202,267	26	149
BPI 2018	43,809	2,514,266	41	165
BPI 2019	251,734	1,595,923	42	627

Fig. 4, 5 represent the experimental results of discovering redo-activities in the BPI 2012 and BPI 2017 datasets, respectively. For the BPI 2012 dataset, totally 6 redo-activities are discovered. The activity "*W_Beoordelen fraude*" has the lowest number of self-repetition (215 times), whereas the activity "*W_Completeren aanvraag*" has the highest number of self-repetition (30,116 times). For the BPI 2017 dataset, totally 13 redo-activities are discovered. The activity "*O_Returned*" has the lowest number of self-repetition (16 times), whereas the activity "*W_Validate application*" has the highest number of self-repetition (115,590 times).

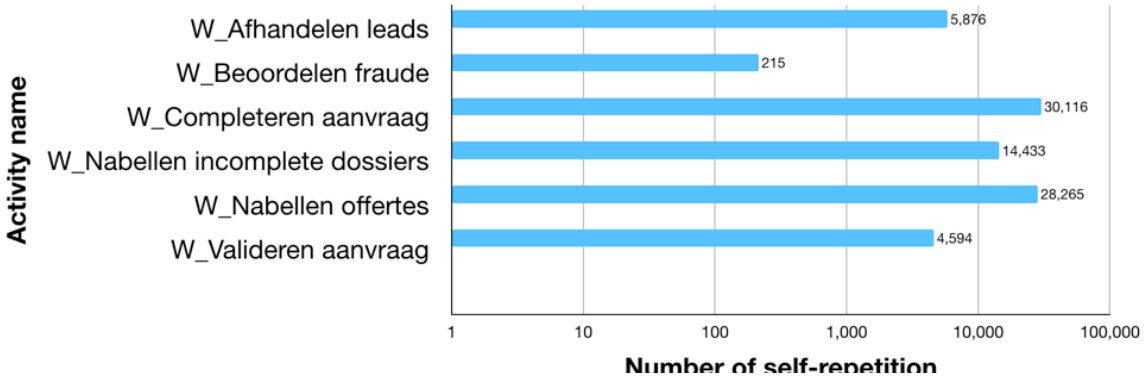


Fig. 4. Occurrences of the redo-activities in the BPI 2012 dataset.

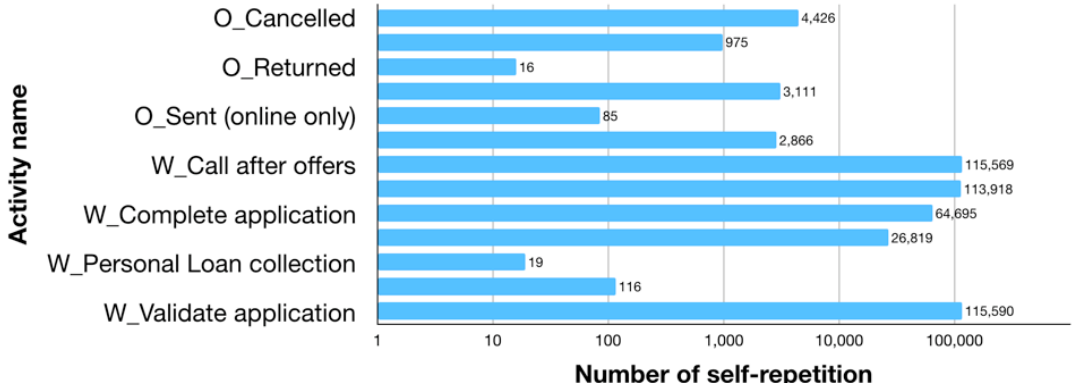


Fig. 5. Occurrences of the redo-activities in the BPI 2017 dataset.

Fig. 6, 7 represent the experimental results of discovering redo-activities in the BPI 2018 and BPI 2019 datasets, respectively. With the BPI 2018 dataset, totally 24 redo-activities are discovered. The activity "*begin editing*" and "*begin preparations*" have the lowest number of self-repetition (2 times), whereas the activity "*save*" has the highest number of self-repetition (79,775 times). With the BPI 2019 dataset, totally 21 redo-activities are discovered. The activity "*Change Rejection Indicator*" and "*Delete Purchase Order Item*" have the lowest number of self-repetition (1 time), whereas the activity "*Record Goods Receipt*" has the highest number of self-repetition (115,590 times).

In addition, we do not consider the information of performers "NONE" and "0;n/a" in the experiments. We also skip the redo-activities cases with the execution time of zero. **Fig. 8** shows the comparisons between redo-activities and the total activities on the datasets. More precisely, the BPI 2018 dataset shows the largest redo-activity ratio (58.54%) while the BPI 2012 dataset has the lowest redo-activity ratio (25%). The comparison is based only on the number of occurrences of redo-activities in each dataset. To explore in more detail, we consider the involvements of performers and the time consumptions of performing redo-activities in section 4.

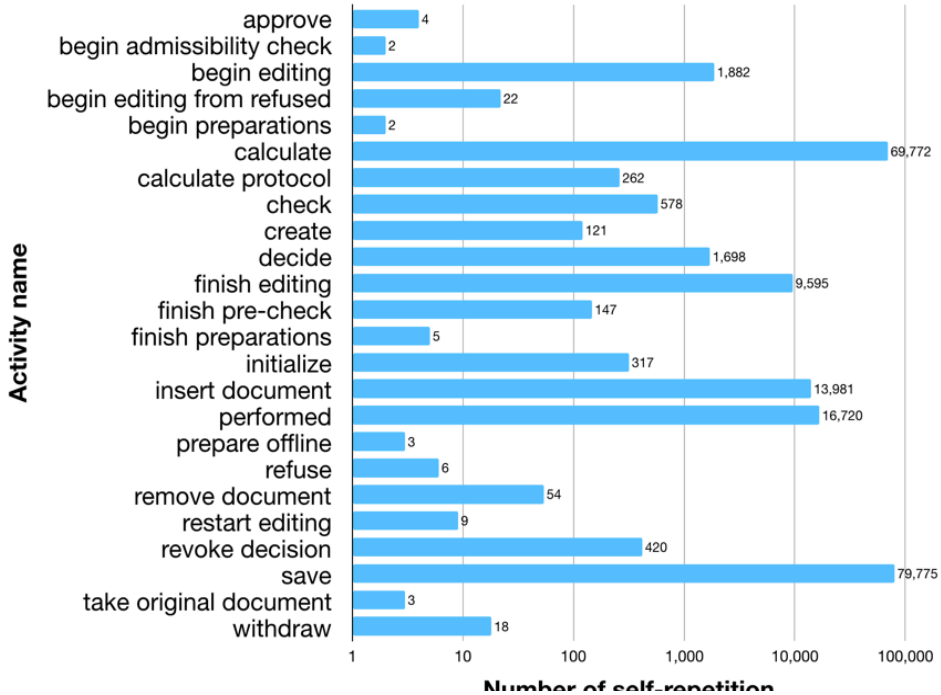


Fig. 6. Occurrences of the redo-activities in the BPI 2018 dataset.

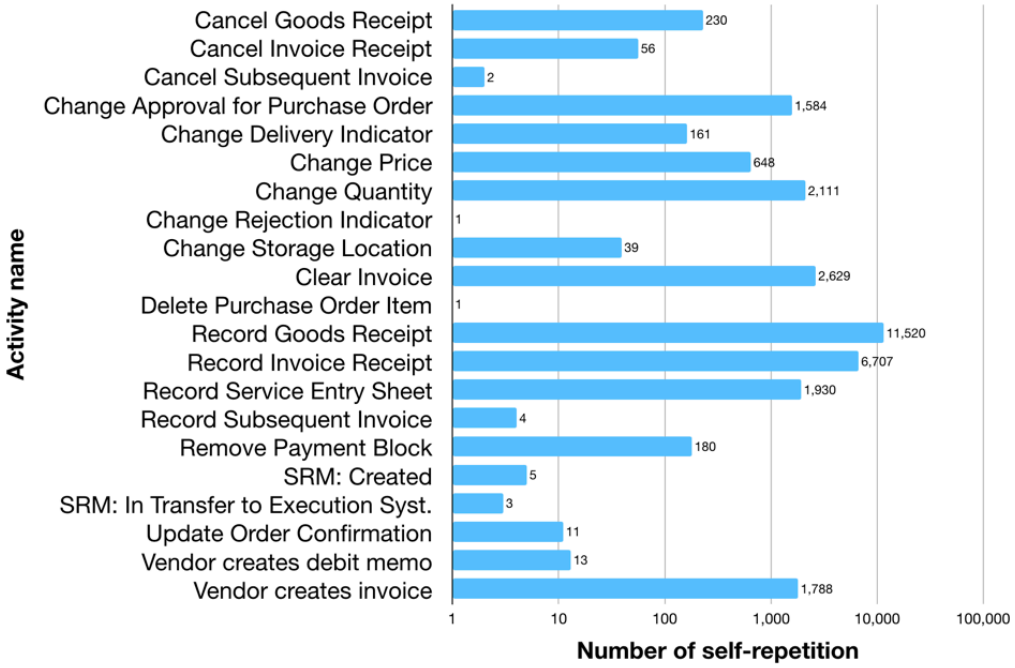


Fig. 7. Occurrences of the redo-activities in the BPI 2019 dataset.

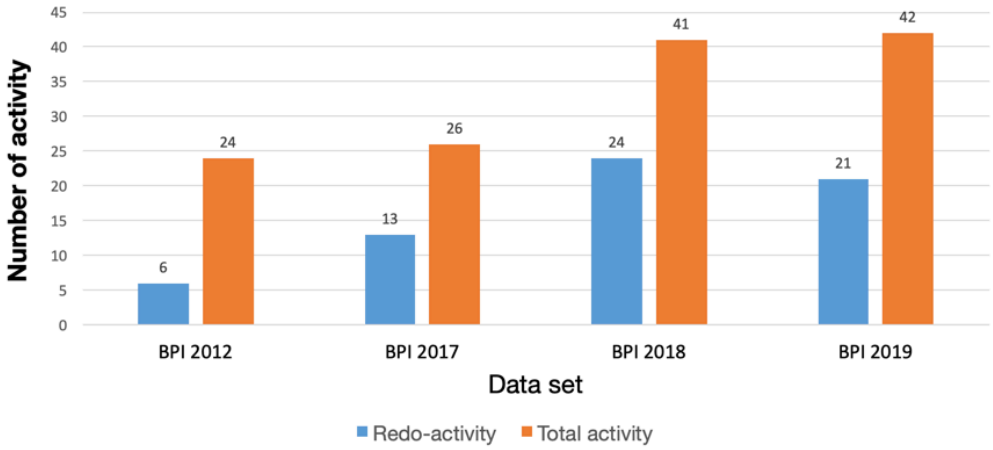


Fig. 8. Comparisons of the redo-activities and total activities.

3.2 The WDG-to-ICN Model Transformation

As mentioned before, redo-activities can be discovered based on the weighted values of arcs of self-loops in the WDG. To express primitive control-flows of workflow processes, the discovered WDG is transformed into an ICN model by the following algorithm. In a transformed ICN model, redo-activities are represented by Loop control-flows. Without loss of generality, we call these Loop control-flows are Loop gate. A Loop-Open gate is the starting point of redo-activity while a Loop-Close gate is the termination of redo-activity. Hence, a Loop-Open gate connects redo-activities with predecessor-activities and a Loop-Close gate connects redo-activity with successor-activities.

Algorithm: WDG-to-ICN Transformation

Input: A weighted directed graph (adjacency list β).
Output: An ICN model with redo-activities
Begin
1: **For**(∀ vertex[i] ∈ (β))
2: currentVertex = vertex[i]
3: **If**(arc(currentVertex,currentVertex) > 0)
4: CreateNewLoop-Open(LO)
5: CreateArc(LO,currentVertex)
6: CreateArc (predecessorOfCurrentVertex, LO)
7: CreateNewLoop-Close(LC)
8: CreateArc (currentVertex, LC)
9: CreateArc (LC, successorOfCurrentVertex)
10: CreateArc(LC,LO)
11: DeleteArc(currentVertex,currentVertex)
12: **Endif**
13: **EndFor**
End

Fig. 9 depicts the redo-activities of **Fig. 3** visualized in ICN model. We can see that three redo-activities (visualized as orange circles) are connected by Loop-Open and Loop-Close gates (visualized as yellow circles).

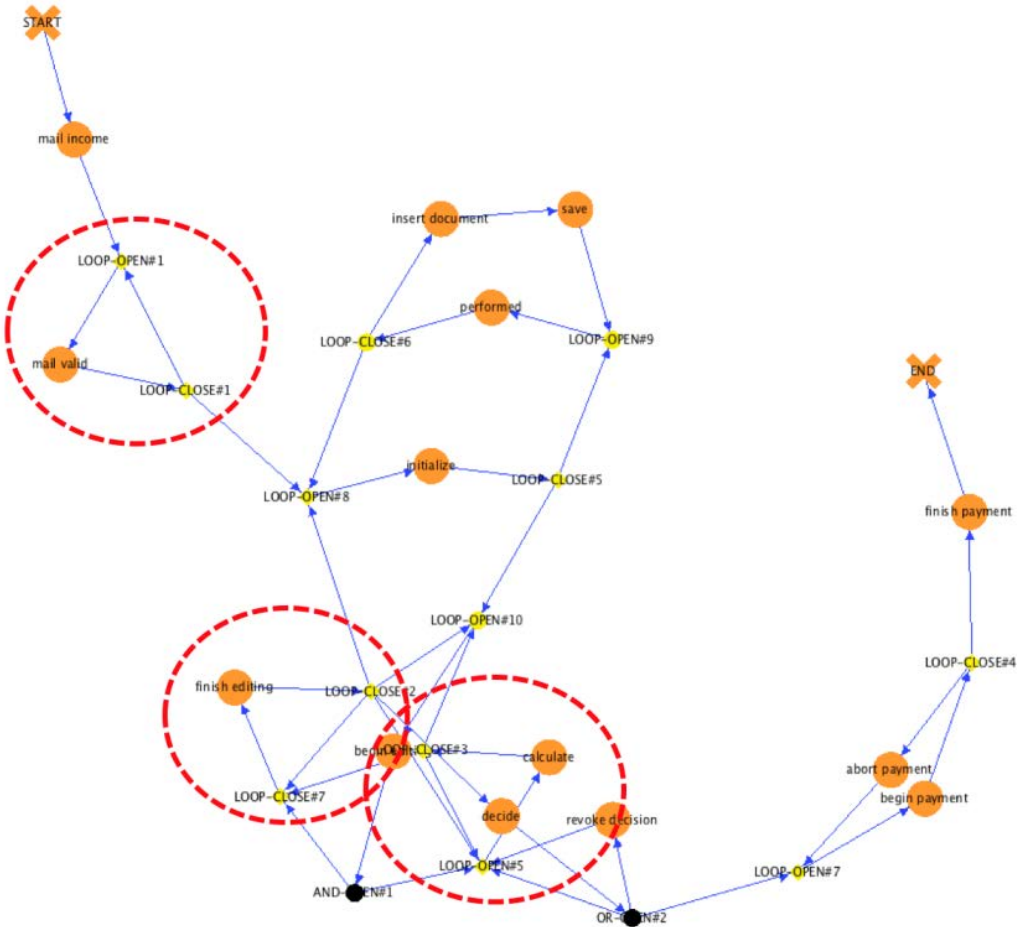


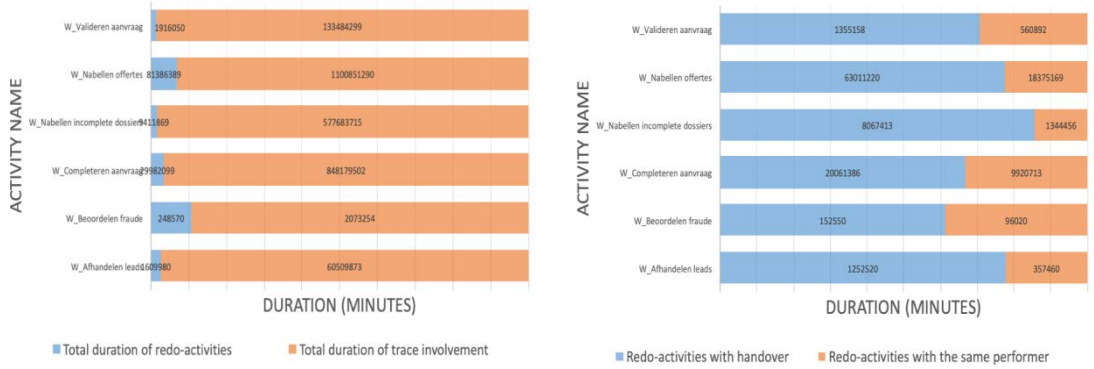
Fig. 9. ICN model of redo-activities in trace number-3.

4. Performers' Involvements and Execution Time in Redo-Activities

In this section, we analyze the relevance of performer involvements in redo-activities and those execution times. To collect and analyze information, we re-explore the event log. With the redo-activities (found by the algorithm described in section 3), we store details about performers and execution times to perform those activities. Then proceed to aggregate statistics about the total execution times of the redo-activities in each trace, in the traces that those redo-activities participated. In discovered redo-activities, we also carry out classify performers (handover or same performer) to compare their execution duration. Some issues need to address when doing this study to avoid incorrect statistics:

- A redo-activity can participate in many different traces hence the execution time should be based on Trace-id and Redo-activity-id.
- In the same trace, performers can both participate in redo-activities and join in performing other activities. For this reason, needing to calculate the execution time by subtotal the Performer-id and Activity-id.

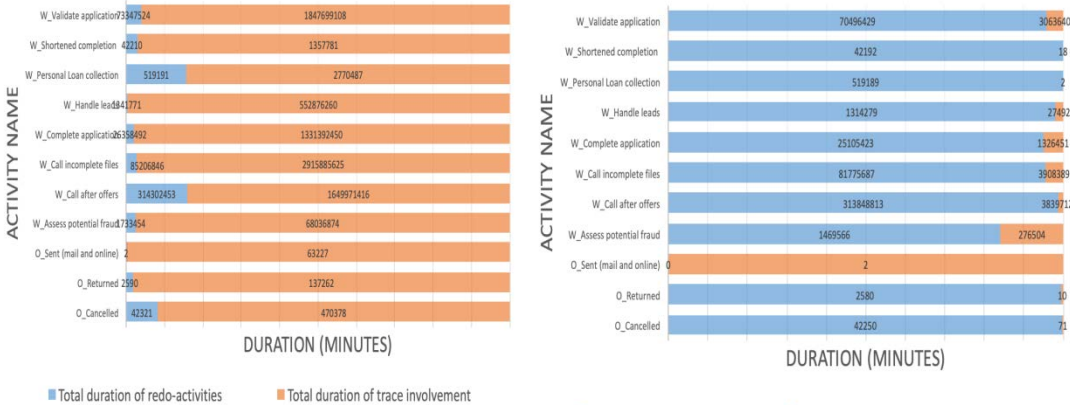
Fig. 10 represents the comparison in the BPI 2012 dataset. In **Fig. 10 (a)**, we compare the execution time of the redo-activities with the total time to complete traces that it related. The redo-activity "*W_Beoordelen fraude*" has the highest execution time ratio (10.70%) while the redo-activity "*W_Nabellen incomplete dossiers*" has the lowest execution time ratio (1.60%). **Fig. 10 (b)** shows the result of comparing performers' involvements types in redo-activities. A redo-activity can be done by either only one performer (the same performer) or two or more performers transferring its control from one performer to the others (handover) in iterations. The redo-activity "*W_Nabellen incomplete dossiers*" has the highest handover ratio (85.71%) while the redo-activity "*W_Beoordelen fraude*" has the lowest handover ratio (61.37%).



(a) (b)

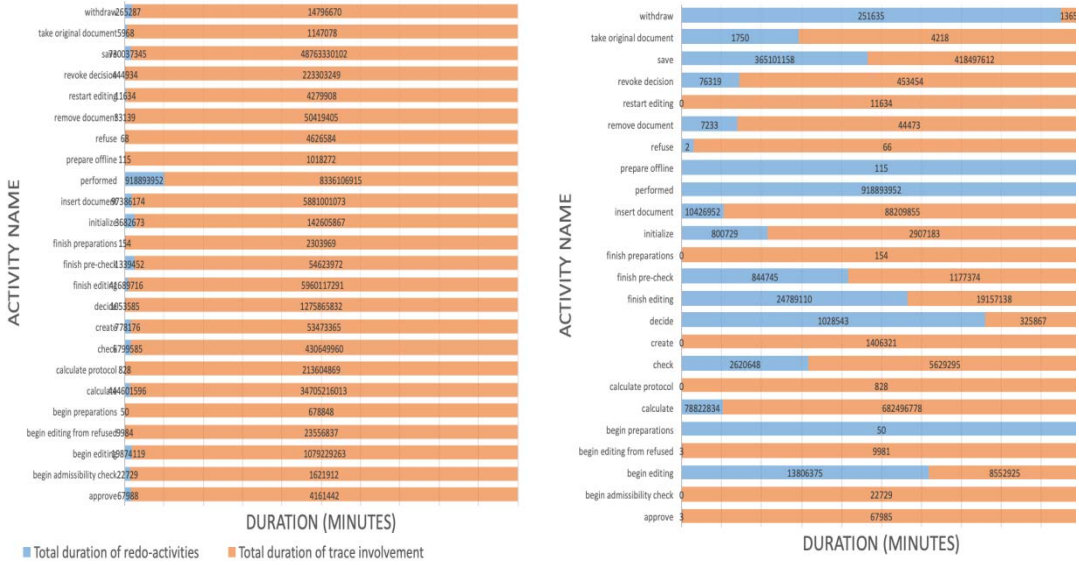
Fig. 10. The comparison of redo-activities with performers' involvements in BPI 2012.

Similarly, **Fig. 11-13** represent the comparison results on the BPI 2017, BPI 2018, and BPI 2019 datasets. As a result, the redo-activities consume certain amounts of time during the execution of a trace. Redo-activities with substantial consumption time are potential activities to cause a bottleneck in process-aware information systems. Redo-activities are usually performed by different performers (handover) more than the same performer.



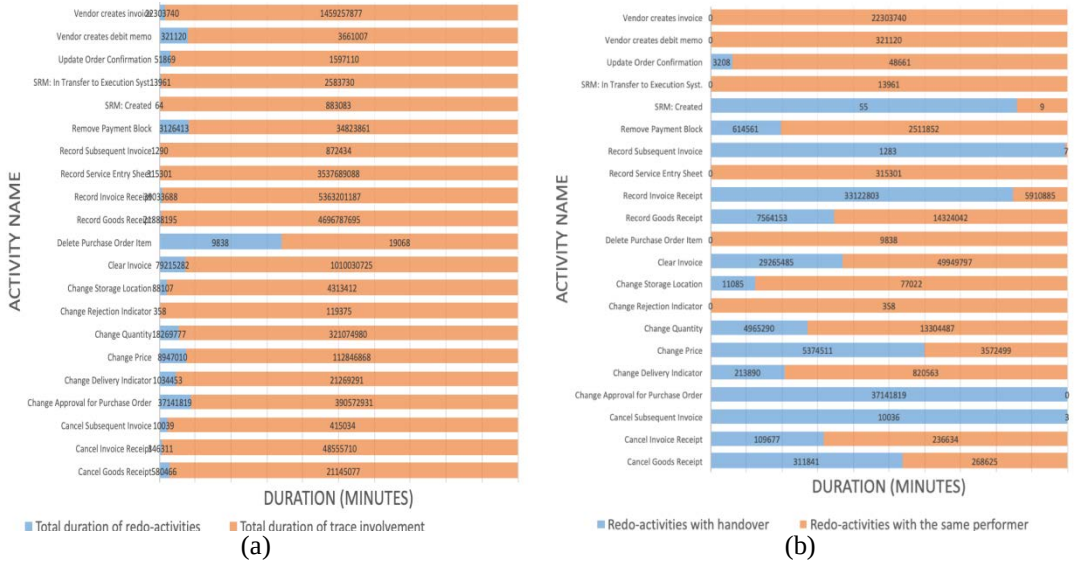
(a) (b)

Fig. 11. The comparison of redo-activities with performers' involvements in BPI 2017.



(a) (b)

Fig. 12. The comparison of redo-activities with performers' involvements in BPI 2018.



(a) (b)

Fig. 13. The comparison of redo-activities with performers' involvements in BPI 2019.

5. Conclusion

In this paper, we proposed and implemented a couple of algorithms for discovering and visualizing those redo-activities and their performers' involvements patterns from a dataset of workflow process enactment event logs formatted in the IEEE XES event stream standard format, and carried out a series of experimental analyses by applying the implemented algorithms to four datasets released from the BPI Challenges (BPI 2012, BPI 2017, BPI 2018, and BPI 2019). Consequently, our works in this paper are summarized as follows:

1) The devised discovery algorithm perfectly works in exploring the iterative process pattern of redo-activities from an XES-formatted dataset and producing its weighted directed graph (WDG) model.
2) The algorithm transforming from the discovered WDG into an information control net model for visualizing the redo-activities in the corresponding workflow process model.
3) The implication of discovering the redo-activities ought to be a valuable knowledge mining activity in terms of analyzing not only the bottleneck points of workflow process models but also the performers' involvements patterns in performing the repetitive redo-activities, like a *single performer involvement pattern* and a multiple performers involvement pattern with calculating work-*handover ratings*. Especially, we would strongly believe that the analytical results of the work-handover rates in performing the repetitive redo-activities be very valuable knowledge to find out those potential activities that can be the sources to be evolving to the system's bottlenecks.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (Grant No.2017R1A2B2010697). This work also was supported by Kyonggi University's Graduate Research Assistantship 2019.

References

[1] P. Hornix, "Performance analysis of business processes through process mining," *Master's Thesis, Eindhoven Univ. Technol.*, no. January, p. 108, 2007. Article (CrossRef Link)
[2] W. Premchaiswadi and P. Porouhan, "Process modeling and bottleneck mining in online peer-review systems," *Springerplus*, vol. 4, no. 1, p. 441, Dec. 2015. Article (CrossRef Link)
[3] K. Kim and C. A. Ellis, "ICN-based Workflow Model and its Advances," *Handb. Res. Bus. Process Model.*, pp. 142–171, 2009. Article (CrossRef Link)
[4] W. M. P. van der Aalst, T. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Trans. Knowl. Data Eng. 9*, vol. 16, no. 9, pp. 1128–1142, 2004. Article (CrossRef Link)
[5] A. J. M. M. Weijters, W. M. P. Van Der Aalst, and A. K. Alves De Medeiros, "Process Mining with the HeuristicsMiner Algorithm," *Tech. Univ. Eindhoven, Tech. Rep. WP*, no. 166, pp. 1–34, 2006. Article (CrossRef Link)
[6] Günther, C. W., and W. M. Van Der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics.," in *Proc. of Int. Conf. Bus. Process Manag. Springer, Berlin, Heidelb.*, pp. 328–343, 2007. Article (CrossRef Link)
[7] A. K. A. De Medeiros, · A J M M Weijters, and · W M P Van Der Aalst, "Genetic process mining: an experimental evaluation," *Data Min Knowl Disc*, vol. 14, pp. 245–304, 2007. Article (CrossRef Link)
[8] W. M. P. van der Aalst, "Process Mining in the Large : A Tutorial," *Eur. Bus. Intell. Summer Sch. Springer, Cham*, vol. 3, pp. 33–76, 2014. Article (CrossRef Link)
[9] M. Gupta, A. Sureka, "Nirikshan: Mining bug report history for discovering process maps, inefficiencies and inconsistencies," in *Proc. of ISEC '14 Proceedings of the 7th India Software Engineering Conference*, 2014. Article (CrossRef Link)

[10] I. Mukhlash, W. N. Rumana, D. Adzkiya, and R. Sarno, "Business process improvement of production systems using coloured petri nets," *Bull. Electr. Eng. Informatics*, vol. 7, no. 1, pp. 102–112, 2018. Article (CrossRef Link)
[11] University of Le Havre, "GraphStream library," http://graphstream-project.org/.
[12] 4TU.Centre for Research Data, "IEEE Task Force on Process Mining - Event Logs," https://data.4tu.nl/repository/collection:event_logs.

Dinh-Lam Pham is a full-time PhD. student in Kyonggi University, South Korea since March, 2018. He got the MSc in computer science, Thai Nguyen University, Vietnam (2010). He worked in Thainguyen University from 2008 to 2016 as an IT engineering. From 2016, he works in the Information Technology Institute, Vietnam National University, Hanoi. His research interests include Process Mining, Workflow Management System, Information Control Net.

Hyun Ahn is an assistant professor of computer science and engineering department at Kyonggi University, South Korea. He received his B.S., M.S., and Ph.D. degrees in computer science from Kyonggi University in 2011, 2013, and 2017, respectively. His research interests include workflow systems, BPM, business process intelligence, and process mining.

Kwanghoon Pio Kim is a full professor of computer science department and the founder and supervisor of the collaboration technology research laboratory at Kyonggi University, South Korea. He received B.S. degree in computer science from Kyonggi University in 1984. And he received M.S. degree in computer science from Chungang University in 1986. He also received his M.S. and Ph.D. degrees from the computer science department at University of Colorado Boulder, in 1994 and 1998, respectively. At Kyonggi University, he is Dean of the Computerization and Informatics Institute, and the director of the contents convergence software research center, as well. He had worked as researcher and developer at Aztek Engineering, American Educational Products Inc., and IBM in USA, as well as at Electronics and Telecommunications Research Institute (ETRI) in South Korea. In present, he is a vice-chair of the BPM Korea Forum. He has been in charge of a country-chair (Korea) and ERC vice-chair of the Workflow Management Coalition. He has also been on the editorial board of the journal of KSII, and the committee member of the several conferences and workshops. His research interests include groupware, workflow systems, BPM, CSCW, collaboration theory, Grid/P2P distributed systems, process warehousing and mining, workflow-supported social networks discovery and analysis, process-aware information systems, data intensive workflows, and process-aware Internet of Things.