

Complex Control-Flow Constructs Detection from Process Related Data

Hind R'bigui*, Chiwoon Cho*

*School of Industrial Engineering, University of Ulsan, Ulsan 680-749, Republic of Korea

hind.rbigui@gmail.com, chiwoon6@mail.ulsan.ac.kr

Abstract— Process mining is new techniques whereby knowledge from event log stored in today's information systems are extracted to automatically construct business process models to have a full understanding of the real behaviour of processes, identify bottlenecks, and then improve them. Many process discovery algorithms have been proposed today. However, there are complex control-flow constructs that current discovery techniques cannot correctly discover in models based on event logs. These constructs are short loops, invisible tasks, and non-free choice constructs. There is currently no algorithm that can handle all of these structures in a restricted time. In this paper, we propose a framework that detects from event logs the complex control-flow constructs that exist. By identifying the existing constructs from a given event log, one can identify the process model notation or the process discovery algorithm appropriate for the given event log. The framework has been implemented in ProM and the results show that constructs are identified correctly.

Keywords— Process mining, process discovery, control-flow, constructs, event log, log characteristics, process structure.

I. INTRODUCTION

Process mining is a recent set of techniques which fall between data mining, machine learning, process modelling and analysis. Process mining consists of extracting knowledge from process related data stored in today's information systems such as Workflow Management Systems, Enterprise Resource Planning, Business Process Management Systems, etc. The business process related data stored in the information systems are called event logs. Using these event logs, three major categories of process mining can be applied [1]. 1) Process discovery techniques takes an event log as input and automatically produce a process model which represents the real behaviour of a business process. 2) Conformance verification techniques compare the model discovered using process discovery techniques with the reference model (i.e., the a-priori model) of an organization or compare the discovered model with the corresponding event log. This allows businesses to confirm or correct their idea regarding what they think is happening by discovering what is really happening. 3) Process enhancement techniques tend to improve or extend the existing model with the information in the discovered model or in the event log. An overview on

process mining framework is illustrated in Figure 1. In the process discovery category, three perspectives are possible depending on the information available in the event log. The control-flow perspective can be discovered if the log provides information about the tasks executed to handle a particular case (i.e., process instance). The organisational perspective can be derived if the event log contains information on the persons, systems, or machines involved in the execution of the tasks. The case perspective can be mined if the event log contains additional data associated with tasks.

In this paper, we focus on the control-flow perspective of process discovery category. There are actually plenty of process discovery techniques that have been developed [2-4] and applied successfully to various fields [5-7]. Nevertheless, there are complex control flow constructs that current discovery algorithms cannot correctly discover in models based on event logs. These constructs are short loops, invisible tasks, and non-free choice constructs. There is currently no algorithm that can handle all of these structures in a restricted time [4], [8]. Moreover, there are several process modelling notations such as BPMN [9], Petri Net [10], Process Tree [11]. These notations can allow some control flow constructs as well as it cannot allow some others.

This work proposes a framework for detecting the complex control flow constructs based only on event logs. The framework can be used to recommend process discovery techniques suitable to a given event log or to recommend a process model notation appropriate to a given event log.

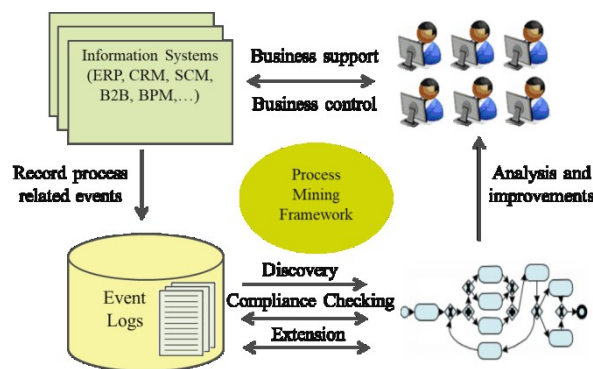


Figure 1. Process mining framework

II. CONTROL FLOW CONSTRUCTS

The most relevant constructs typically discovered by process discovery algorithms are sequence, exclusive choice, inclusive choice, parallelism, loops, invisible tasks, and non-free choice constructs. We classify these constructs into standards constructs that can be discovered by all of existing process discovery algorithms and complex constructs that cannot be mined by all of the algorithms. We are not concerned with the standards constructs. Our main focus, in this paper, is on the complex constructs which are short loops, invisible tasks, and non-free choice constructs.

A. Short Loops

Short loops are classified into a short loop of length one and or a short loop of length two. In a loop of length one, there is one task that can be repeated several times, while a loop of length two involves two tasks that can be repeated many times.

B. Invisible Tasks

An invisible task is a hidden task that has been executed but does not appear in the event log. For instance, the execution of a task that has been skipped to allow flexibility in the execution of a process is an invisible task that exists in the process model but cannot be recorded in the event log [12]. We classify invisible tasks into Short Skip in sequence, Short Skip in parallel, Long Skip in sequence and Long Skip in Parallel to skip the execution of one or multiple activities; and into Short Redo in sequence, Short Redo in parallel, Long Redo in sequence and Long Redo in Parallel to repeat the execution of one or multiple activities.

C. Non-Free Choice Constructs

A non-free choice construct combines choice and synchronization [13] which means, the choice between two or more activities is dependent to the activities that have been executed before in a given process model. Non-free choice constructs can be detected by detecting implicit dependencies. Implicit dependency is an indirect causal relationship between activities.

III. COMPLEX CONTROL-FLOW CONSTRUCTS DETECTION

In this section, we present the equations used to detect short loops, invisible tasks, non-free choice and non-free choice involved in invisible tasks in a given event log. The equations for detecting the complex constructs are based on the extensions of alpha miner [2]. Compared to other algorithms, the extensions of alpha miner (i.e., α^{++} [13], α^s [14], $\alpha^\#$ [12]) address most of the construct separately, i.e., short loops, all types of invisible tasks and non-free choice constructs. The ordering relations defined in alpha algorithm extensions are used to discover a process model. In this paper, we use the ordering relations and improve some of them to detect the existence of complex control-flow constructs in an event log.

A. Basic and Advanced Ordering Relations

In α algorithm, the originator and the driver of the α – extensions algorithms, the following relations were defined: $>_L$, Δ_L , $\diamond_L$, $\rightarrow_L$, $\parallel_L$, $\neq_L$, $\gg_L$, $\succ_L$, $\triangleleft_L$, $\triangleright_L$, $\triangleleft_L$, Between-Set, as $\Rightarrow_L$ and $\geq_L$. Relation $>_L$ represents two activities that can successively be executed. If the same activity is executed two or multiple times successively, then the direct succession refers to a loop with length-one structure. Relation Δ_L expresses a loop with length-two structure (e.g., *aba*). This relation is used also to distinguish length-2-loop from a parallel routing. Relation $\diamond_L$ refers to two-way loop-2-length, which means two activities have the Δ_L relations between each other (e.g. *aba*, *bab*). Relation $\rightarrow_L$ shows the direct causal relation between two activities. Relation $\parallel_L$ represents the activities that can be executed concurrently (e.g., *ab*, *ba*). Relation $\neq_L$ refers to two activities never following each other directly. $\gg_L$ reflects the indirect reachable dependency between two tasks and $\succ_L$ reflects the indirect or direct reachable dependency between two tasks. Relation $\triangleleft_L$ represents XOR-Split, $\triangleright_L$ corresponds to XOR-Join and $\triangleleft_L$ represent AND-split. Between-Set (*Between(L, a, b)*) depict the activities that occur between two activities. If the two tasks are the endpoints of a parallel construct, the Between-Set is the set of activities in the parallel branches. In a parallel workflow, the relations $\rightarrow_L$ and $>_L$ are defined as $\Rightarrow_L$ and $\geq_L$ respectively to eliminate the interference of parallel constructs ($\Rightarrow_L^{sh}$ and $\geq_L^{sh}$ for short skip and redo, $\Rightarrow_L^{sh}$ and $\geq_L^{sh}$ for long skip and redo).

Definition (Basic and advanced Ordering relations)

Let A be a set of activities, L be an event log over A , a and b be two activities in A , the ordering relations defined in α and α^+ , α^{++} , $\alpha^\#$, α^s algorithms as follows:

- $a >_L b \Leftrightarrow \exists \sigma = t_1 t_2 \dots t_n \in L, i \in 1, \dots, n - 1: t_i = a \wedge t_{i+1} = b,$
- $a \Delta_L b \Leftrightarrow \exists \sigma = t_1 t_2 \dots t_n \in L, i \in 1, \dots, n - 2: t_i = t_{i+2} = a \wedge t_{i+1} = b,$
- $a \diamond_L b \Leftrightarrow (a \Delta_L b) \vee (b \Delta_L a),$
- $a \rightarrow_L b \Leftrightarrow ((a >_L b) \wedge (b \neq_L a)) \vee (a \diamond_L b),$
- $a \parallel_L b \Leftrightarrow (a >_L b) \wedge (b >_L a) \wedge (b \neq_L a),$
- $a \neq_L b \Leftrightarrow ((a \neq_L b) \wedge (b \neq_L a)),$
- $a \gg_L b \Leftrightarrow \exists \sigma = t_1 t_2 \dots t_n \wedge i, j \in 1, \dots, n: i < j \wedge t_i = a \wedge t_j = b \wedge \forall k \in [i + 1, \dots, j - 1]: t_k \neq a \wedge t_k \neq b,$
- $a \succ_L b \Leftrightarrow (a \rightarrow_L b) \vee (a \gg_L b),$
- $a \triangleleft_L b \Leftrightarrow (a \neq_L b) \wedge \exists c \in A: (c \rightarrow_L a) \wedge (c \rightarrow_L b),$
- $a \triangleright_L b \Leftrightarrow (a \neq_L b) \wedge \exists c \in A: (a \rightarrow_L c) \wedge (b \rightarrow_L c),$
- $a \triangleleft_L \Leftrightarrow \exists x, y \in A: (a \rightarrow_L x) \wedge (a \rightarrow_L y) \wedge (x \parallel_L y),$
- $Between(\sigma, a, b) = \{\sigma_k | \exists 1 \leq i < j \leq m (\sigma_i = a \wedge \sigma_j =$

- $b) \wedge i < k < j \wedge \nexists_{i < l < j} (\sigma_l = a \vee \sigma_l = b))\}$,
- $\neg \text{Between}(\sigma, a, b) = \{\sigma_k | 1 \leq k \leq m\} \setminus \text{Between}(\sigma, a, b)$,
- $\text{Between}(L, a, b) = \cup_{\sigma \in L} \text{Between}(\sigma, a, b) \setminus \cup_{\sigma \in L} \neg \text{Between}(\sigma, a, b)$,
- $a \geq_L^{sh} b \Leftrightarrow \exists x \in \text{Between}(L, a, b) \wedge \forall m \in (\text{Between}(L, a, b) \setminus \{x\}): (m \parallel_L x) \wedge \exists_{\sigma \in L} \text{Between}(\sigma, a, b) \subseteq (\text{Between}(L, a, b) \setminus \{x\})$,
- $a \geq_L^{sh} b \Leftrightarrow ((a \geq_L^{sh} b) \wedge (b \not\geq_L^{sh} a)) \vee (a \not\geq_L b)$,
- $a \geq_L^{lg} b \Leftrightarrow \exists_{x, y \in A} \text{Between}(L, x, y) \subseteq \text{Between}(L, a, b) \wedge \forall m \in (\text{Between}(L, a, b) \setminus (\text{Between}(L, x, y) \cup \{x, y\})) \forall n \in \text{Between}(L, x, y): (m \parallel_L n) \wedge \exists_{\sigma \in L} \text{Between}(\sigma, a, b) \subseteq (\text{Between}(L, a, b) \setminus \{x, y\})$.

B. Control-Flow Constructs Detecting Dependencies

In this section, we present the dependencies that detects the complex control-flow constructs: short loops of length one and two, invisible tasks of types short skip and redo and long skip and redo in sequence and in parallel, non-free choice constructs, and invisible tasks involved in a non-free choice constructs. The dependencies are defined in Table 1.

TABLE 1. CONTROL-FLOW CONSTRUCTS DETECTING DEPENDENCIES

Constructs	Detecting equation
Loop of length one	$a >_L a \Leftrightarrow \exists \sigma = t_1 t_2 \dots t_n \in L, i \in 1, \dots, n-1: t_i = a \wedge t_{i+1} = a$.
Short skip in sequence	$a \rightarrow_L^{SSS} b \Leftrightarrow (a \rightarrow_L b) \wedge \exists x \in A: (a \rightarrow_L x) \wedge (x \rightarrow_L b) \wedge (x \not\parallel_L b) \wedge (a \not\parallel_L x) \wedge (x \not\geq_L x)$.
Short redo in sequence	$y \rightarrow_L^{SRS} x \Leftrightarrow (y \rightarrow_L x) \wedge \exists a \in T: (a \rightarrow_L x) \wedge (y \rightarrow_L a) \wedge (x \not\parallel_L a) \wedge (a \not\parallel_L y) \wedge (a >_L a)$.
Long skip in sequence	$a \rightarrow_L^{LSS} b \Leftrightarrow (a \rightarrow_L b) \wedge \exists x, y \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L b) \wedge (y \not\geq_L x) \wedge (x \not\parallel_L b) \wedge (a \not\parallel_L y) \wedge (x \not\geq_L y)$.
Long redo in sequence	$y \rightarrow_L^{LRS} x \Leftrightarrow \exists a, b \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L b) \wedge (x \not\parallel_L b) \wedge (y \not\parallel_L a) \wedge (b \not\geq_L a) \wedge (a >_L b)$.
Short skip in parallel	$a \rightarrow_L^{SSP} b \Leftrightarrow \exists x, y \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L b) \wedge (x \not\parallel_L b) \wedge (y \not\parallel_L a) \wedge (a \not\geq_L^{sh} b)$.
Short redo in parallel	$y \rightarrow_L^{SRP} x \Leftrightarrow \exists a \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L a) \wedge (a \not\parallel_L x) \wedge (y \not\parallel_L a) \wedge (y \not\geq_L^{sh} x) \wedge (a >_L a)$.
Long skip in parallel	$a \rightarrow_L^{LSP} b \Leftrightarrow \exists x, y \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L b) \wedge (x \not\parallel_L b) \wedge (y \not\parallel_L a) \wedge (a \not\geq_L^{lg} b) \wedge (y \not\geq_L x) \wedge (x \not\geq_L y)$.
Long redo in parallel	$y \rightarrow_L^{LRP} x \Leftrightarrow \exists a, b \in A: (a \rightarrow_L x) \wedge (y \rightarrow_L b) \wedge (x \not\parallel_L b) \wedge (y \not\parallel_L a) \wedge (y \not\geq_L^{lg} x) \wedge (b \not\geq_L a) \wedge (b \not\geq_L x)$.
Non-free Choice Constructs	$a \rightarrow_{L^1} b \Leftrightarrow (a \not\geq_L b) \wedge \exists c \in A: (a >_L c) \wedge (c <_L b) \wedge \nexists t \in A: (t >_L a) \wedge ((t >_L a) \vee (t \parallel_L a))$, $a \rightarrow_{L^{21}} b \Leftrightarrow (a \gg_L b) \wedge a <_L a \wedge \exists b' \in A: (b <_L b') \wedge \nexists t \in A: a >_L t \wedge (t \not\geq_L b \text{ or } t \parallel_L b) \wedge \exists t' \in A: a >_L t' \wedge (t' \not\geq_L b \vee t' \parallel_L b)$, $a \rightarrow_{L^{22}} b \Leftrightarrow (a \gg_L b) \wedge b <_L a \wedge \exists a'' \in A: (a >_L a') \wedge \nexists t \in A: t >_L b \wedge (a \not\geq_L t \text{ or } a \parallel_L t) \wedge \exists t' \in A: t' >_L b \wedge (a' \not\geq_L t' \vee a' \parallel_L t')$, $a \rightarrow_{L^2} b \Leftrightarrow a \rightarrow_{L^{21}} b \vee a \rightarrow_{L^{22}} b$, and

	$a \rightarrow_{L^3} b \Leftrightarrow (a <_L a') \wedge (b >_L b') \wedge (a \gg_L b) \wedge (a \not\gg_L b')$.
Invisible tasks involved in a non-free choice construct	$x \rightarrow_L m \Leftrightarrow \exists_{(a=\perp \vee a \in L) \wedge b \in L} (a \rightarrow_L x) \wedge (x \rightarrow_L b) \wedge b \not\geq_{L, pre=a} m \wedge \nexists_{n \in \sigma} b \not\geq_{L, pre=a} n$, $m \rightarrow_L x \Leftrightarrow \exists_{a \in L} \wedge (b \in L \vee b = \top) (a \rightarrow_L x) \wedge (x \rightarrow_L b) \wedge m \not\geq_{L, pre=b} a \wedge \nexists_{n \in \sigma} m \not\geq_{L, pre=n} a$, $x \rightarrow_L y \Leftrightarrow \exists_{(a_1=\perp \vee a_1 \in L) \wedge b_1 \in L \wedge a_2 \in L \wedge (b_2 \in L \vee b_2 = \top)} (a_1 \rightarrow_L x) \wedge (x \rightarrow_L b_1) \wedge (a_2 \rightarrow_L y) \wedge (y \rightarrow_L b_2) \wedge b_1 \not\geq_{L, pre=a_1, post=b_2} a_2$ $\wedge \nexists_{n \in L} b_1 \not\geq_{L, pre=a_1, post=b_n} a_2$.

Dependency $a >_L a$ detect short loop of length one. The following dependencies $a \rightarrow_L^{SSS} b$, $x \rightarrow_L^{SRS} y$, $a \rightarrow_L^{LSS} b$, $x \rightarrow_L^{LRS} y$, $a \rightarrow_L^{SSP} b$, $y \rightarrow_L^{SRP} x$, $a \rightarrow_L^{LSP} b$, $y \rightarrow_L^{LRP} x$ reflects the mendacious dependencies associated with invisible tasks of types Short Skip in Sequence, Short Redo in Sequence, Long Skip in Sequence, Long Redo in Sequence, Short-Skip in Parallel, Short-Redo in Parallel, Long-Skip in Parallel, Long-Redo in Parallel respectively. The dependencies $\rightarrow_{L^1}$, $\rightarrow_{L^2}$, and $\rightarrow_{L^3}$ are implicit dependencies which detect non-free choice constructs. $x \rightarrow_L m$, $m \rightarrow_L x$, $x \rightarrow_L y$ detect reachable dependencies related to invisible tasks in a non-free choice construct.

IV. IMPLEMENTATION AND EVALUATION

We have implemented the framework detecting complex control flow constructs in ProM. ProM is an extensible framework that provides a comprehensive set of plugins for the discovery, conformance checking and analysis of process models from event logs and can be downloaded from <http://www.processmining.org>. The plugin works as follows. Once the event log in question is imported in ProM, the plugin “Control Flow Constructs Detection” is selected and run. The output is a table which indicates whether the complex constructs (i.e., Short loops of length one (L_{1p} for short), Short loops of length two (L_{2p}), invisible tasks of type Short-Skip in sequence (IVT – SS_{seq}), invisible tasks of type Short-Skip in parallel (IVT – SS_{par}), invisible tasks of type Short-Redo in sequence (IVT – SR_{seq}), invisible tasks of type Short-Redo in parallel (IVT – SR_{par}), invisible tasks of type Long-Skip in sequence (IVT – LS_{seq}), invisible tasks of type Long-Skip in parallel (IVT – LS_{par}), invisible tasks of type Long-Redo in sequence (IVT – LR_{seq}), invisible tasks of type Long-Redo in parallel (IVT – LR_{par}), non-free choice construct (NFC), and invisible tasks involved in a non-free choice construct (IVT-NFC)) exist or not in the event log by showing “YES” if exist and “NO” if not.

The framework has been evaluated using 20 artificial event logs. The 20 artificial event logs contain randomly the aforementioned constructs. In the set of 20 models, the maximum number of activities in a process model is less than 15 and the number of cases in one event log is less than 30. The obtained results on the existing constructs are compared with the constructs existing in the reference models of the 20 logs. The comparison is shown in Table 2.

The results shows that the constructs existing in reference

models and the constructs detected with our framework are similar which indicates that our framework is powerful.

TABLE 2. COMPARISON BETWEEN THE CONSTRUCTS EXISTING IN THE REFERENCE MODELS AND THE CONSTRUCTS DETECTED WITH THE PROPOSED FRAMEWORK

Event logs	Detected constructs	Reference models	Existing constructs
L1	$IvT_{SS_{par}}$	M_{r1}	$IvT_{SS_{par}}$
L2	$IvT_{LR_{par}}$	M_{r2}	$IvT_{LR_{par}}$
L3	$IvT_{SS_{seq}}$	M_{r3}	$IvT_{SS_{seq}}$
L4	$IvT_{LS_{seq}}$	M_{r4}	$IvT_{LS_{seq}}$
L5	$L_{2p}, IvT_{LR_{seq}}$	M_{r5}	$L_{2p}, IvT_{LR_{seq}}$
L6	$L_{1p}, IvT_{LR_{par}}$	M_{r6}	$L_{1p}, IvT_{LR_{par}}$
L7	IvT_{NFC}	M_{r7}	IvT_{NFC}
L8	$NFC, IvT_{SS_{seq}}$	M_{r8}	$NFC, IvT_{SS_{seq}}$
L9	$L_{1p}, IvT_{SS_{seq}}, IvT_{LR_{seq}}$	M_{r9}	$L_{1p}, IvT_{SS_{seq}}, IvT_{LR_{seq}}$
L10	L_{1p}, L_{2p}, NFC	M_{r10}	L_{1p}, L_{2p}, NFC
L11	$IvT_{NFC}, IvT_{SS_{seq}}$	M_{r11}	$IvT_{NFC}, IvT_{SS_{seq}}$
L12	$IvT_{SR_{seq}}, L_{2p}$	M_{r12}	$IvT_{SR_{seq}}, L_{2p}$
L13	$L_{2p}, IvT_{SS_{par}}$	M_{r13}	$L_{2p}, IvT_{SS_{par}}$
L14	$L_{1p}, IvT_{LR_{seq}}, NFC$	M_{r14}	$L_{1p}, IvT_{LR_{seq}}, NFC$
L15	$IvT_{SS_{seq}}, IvT_{LS_{seq}}$	M_{r15}	$IvT_{SS_{seq}}, IvT_{LS_{seq}}$
L16	$L_{2p}, IvT_{LS_{seq}}, IvT_{SS_{seq}}$	M_{r16}	$L_{2p}, IvT_{SS_{seq}}, IvT_{LS_{seq}}$
L17	$IvT_{LS_{seq}}, NFC$	M_{r17}	$IvT_{LS_{seq}}, NFC$
L18	$IvT_{SS_{seq}}, IvT_{LS_{seq}}, L_{2p}, NFC$	M_{r18}	$IvT_{SS_{seq}}, IvT_{LS_{seq}}, L_{2p}, NFC$
L19	L_{2p}, IvT_{NFC}	M_{r19}	L_{2p}, IvT_{NFC}
L20	L_{1p}, IvT_{NFC}	M_{r20}	L_{1p}, IvT_{NFC}

V. CONCLUSIONS

This study proposed a framework to detect the complex control flow constructs that can exist from a given event log. The framework has been implemented in ProM and evaluated using artificial event logs and their reference models. The results show that the constructs detected using the proposed framework are similar to the constructs existing in the reference models. This framework can be used for recommending the appropriate algorithm or the appropriate process model notation to a given event log. In future work, we will evaluate this framework using real-life event log.

ACKNOWLEDGMENT

This work was supported by the National IT Industry Promotion Agency (NIPA) grant funded by the Korea government (MSIP) (S1106-16-1001, the development of the Shipyard IHOP process standard simulator and decision support system).

REFERENCES

- [1] W.M.P. Van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 1st ed., Springer-Verlag Berlin Heidelberg, 2011.
- [2] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs", *IEEE Transactions on Knowledge and Data Engineering*, vol.16, no.9, pp. 1128-1142, 2004.
- [3] S. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour". International Conference on Business Process Management, LNBIP, vol. 171, Cham/Springer, pp. 66-78, 2014.
- [4] H. R'bigui, and C. Cho, "Heuristic Rule-based Process Discovery Approach from Events Data", *International Journal of Technology, Policy and Management, Special Issue on: Knowledge Management Systems*, vol.18, no. 4, in press, 2018.
- [5] H. R'bigui and C. Cho, "Customer Order Fulfillment Process Analysis with Process Mining: An Industrial Application in a Heavy Manufacturing Company", *Proceedings of the ACM International Conference on Computer Science and Artificial Intelligence*, pp. 247-252, Jakarta, Indonesia, Dec 05-07, 2017.
- [6] B. Vazquez Barreiros, D. Chapela, M. Mucientes, M. Lama, and D. Barea, Process Mining in IT Service Management: A Case Study. In *International Workshop on Algorithms & Theories for the Analysis of Event Data* (Toruń, Poland), CEUR-WS.org, pp. 16-30, 2016.
- [7] H. R'bigui and C. Cho, "Purchasing Process Analysis with Process Mining of a Heavy Manufacturing Industry", *IEEE Proceeding of the 9th International Conference on Information and Communication Technology*, pp. 495-498, Jeju Island, Korea, Oct 17-19, 2018.
- [8] H. R'bigui and C. Cho, "The state-of-the-art of business process mining challenges", *International Journal of Business Process Integration and Management*, vol.8, no.4, pp. 285-303, 2017.
- [9] Business Process Model and Notation (BPMN). <http://www.omg.org/spec/BPMN/2.0/>.
- [10] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs", *IEEE Transactions on Knowledge and Data Engineering*, vol.16, no.9, pp. 1128-1142, 2004.
- [11] W.M.P. van der Aalst, J. Buijs, and B.F. van Dongen, "Towards Improving the Representational Bias of Process Mining". In K. Aberer, E. Damiani, and T. Dillon, editors, *IFIP International Symposium on Data-Driven Process Discovery and Analysis*, vol.116, pp. 39-54, 2012.
- [12] L. Wen, J. Wang and J. Sun, "Mining Invisible Tasks from Event Logs", in *Lecture Notes in Computer Science 4505*, Springer, Berlin, Heidelberg, pp. 358-365, 2007.
- [13] L. Wen, W M P. Van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs", *Data Mining and Knowledge Discovery*, vol.15 no.2., pp. 145-180, 2007.
- [14] Q. Guo, L. Wen, J. Wang, Z. Yan and P S. Yu, "Mining invisible tasks in non-free-choice constructs", in *Lecture Notes in Computer Science 9253*, Motahari-Nezhad ., Recker J, Weidlich M (eds.), Springer, Cham, pp. 109-125, 2015.



Hind R'bigui received the State Engineer degree in Industrial Engineering from the National School of Applied Sciences of Fes, Morocco in 2013. Currently, she is a Ph.D. candidate at the School of Industrial Engineering at the University of Ulsan, South Korea. Her research interests include business process management, process mining and big data.



Chiwoon Cho is a full Professor of Industrial Engineering at the University of Ulsan in Korea. He has a lot of industry experiences including Hyundai Heavy Industries, LG CNS, and Samsung SDS. His research interests include business process management, process mining and big data.