Business Process Recovery Based on System Log and Information of Organizational Structure

Ryota Mibe*†, Tadashi Tanaka*, Takashi Kobayashi†, and Shingo Kobayashi‡
*Center for Technology Innovation Systems Engineering, Research & Development Group Hitachi,Ltd.
292 Yoshida-cho, Totsuka-ku, Yokohama-shi, 244-0817 Japan
†School of Computing, Tokyo Institute of Technology. 2-12-1 Ookayama, Meguro-ku Tokyo 152-8550 Japan
‡Japan EXpert Clone Corporation. KDX Shinjuku286 bldg. 9F, 2-8-6 Shinjuku, Shinjuku-ku, Tokyo 160-0022 Japan

***Abstract*—In most current cases of enterprise system development, the requirement specifications should follow those of an existing legacy system. However, it is difficult to identify high-level specifications, such as business process steps, from legacy and undocumented systems. In this paper, we propose a method to recover an abstract business process by using system logs and the organizational information of the operators using an existing legacy system. Our method provides a hierarchical view based on a clustering technique to find abstract activities that consist of a series of operations. We also propose a method to extract the main operation in a cluster. We evaluated the effectiveness of our method through experiments on a real system.**

I. Introduction

Approximately 80 percent of business activities are supported by some kind of software systems. This means that the specifications of existing systems need to be considered in most cases of novel system development. However, few recent systems contain any explicit documentation or models [1], because of which it takes a considerable amount of time to understand them.

To solve this problem, various specification recovery methods have been proposed. We can categorize them into two types: static-analysis based and dynamic analysis-based approaches. Static analysis-based methods recover the domain model and the software architecture [2]–[6] through source code analysis. It has been used for 20 years, and is useful for exhaustively recovering procedure-level specifications. However, it does not recover any specifications of modules that have no source code, such as a library or middleware, or specifications of the relations between functions, like business processes for systems that have multiple functions for a workflow.

On the contrary, dynamic analysis based approaches are based on run-time information, such as operation logs and execution trace data [7]–[9]. These approaches can recover sequence or flow specifications. In general, there is a large amount of run-time information, which leads to issues regarding how to acquire, store, and analyze it efficiently. Moreover, dynamic analysis recovers activity relation from the sequence of operations in system logs. It makes accurate specifications, but the granularity of the activity in the system log is too small to give us the so-called big picture of the business process.

In this paper, we propose a new dynamic analysis based method to recover the big picture of a business process based on system logs and organizational information, which are stored for system maintenance and operation. By analyzing them, when they are acquired in system operation, we can accurately recover an adequate granular abstracted business process.

II. Related Work

The business model mining techniques are a popular set of methods for business process recovery [10]. They can recover business process models from sequences of event, method calls in execution traces, and operations in system logs. The recovered business process model represents all system behavior recorded in system logs and can be used for performance analysis and comprehension of detailed behavior. However, the operation-level model is not fit for observing the big picture of the business process because it consists from a large number of operations and complex relationships among them, when we discuss about them with end users who have less knowledge about detail of computer systems.

To solve this problem, some methods to classify or cluster log entries have been proposed. Weerdt et al. proposed ActiTraC to cluster mixed-function log entries into separate data [11]. They used an active learning-inspired clustering approach to separate the different functions of the log by performing selective sampling on the basis of the distance and frequency of occurrence between traces.

Francescomarino et al. proposed a restoration method for BPMN (Business Process Modeling Notation) models for Web applications [12]. They defined three structural criteria (loop, sequence, and alternative) and a logical criterion (page) for clustering business activities into abstracted activities. They use logs and information about the relations among web pages and activities to cluster partial part of a business process. Weerdt also proposed measures for evaluating the discovered process models [13], [14].

However, real business systems are used in parallel for users to collaboratively do their work. The business process model must describe this collaboration. We propose a method to recover an abstract business process by using system logs and the organizational information of the operators using an existing legacy system. It can recover multi-user collaborative activities at an adequate abstracted level.

978-1-5090-5501-2/17 © 2017 IEEE

SANER 2017, Klagenfurt, Austria
Industrial Research

Time Stump	User ID	Action ID	Case ID
7/1 10:00:00:00	Sato	Entry estimate form	00100
7/1 10:00:01:00	Sato	Entry form error	00100
7/1 10:00:02:00	Sato	Entry estimate form	00100
7/1 10:00:05:00	Sato	Request approval	00100
7/1 10:00:06:00	Koizumi	Entry estimate form	00101
7/1 10:00:06:00	Koizumi	Request estimate approval	00101
7/1 11:00:05:00	Suzuki	Check approval request	00100
7/1 11:00:07:00	Suzuki	Approve	00100
7/1 11:00:05:00	Suzuki	Check approval request	00101
7/1 11:00:07:00	Suzuki	Approve	00101
...	...	...	...

Fig. 1. Sample of system log.

III. Preliminaries

We focused on business application systems as target system. They support business activities related to many users with several roles. Each task of a business case is performed collaboratively by some users. Such systems process artifacts from one role to another. A user sometimes has several roles. Each user belongs to an organization (sales department, stock administration) depending on the role, and the organization has a hierarchical structure.

The system log is a record of usage at the time of system operation. Figure 1 shows an example of information that can be restored from the system log. It contains the Time stamp, User ID, Action ID and Case ID for operations. Case ID shows the artifact that a given operation acted on, for example, registration number, receipt number, and order number.

Organizational Structure Information consists of knowledge of users (identified by user IDs), roles, and the parts of the organizational structure (company, departments, sections, groups, units, etc). They are usually described by a tree structure. Each user must belong to one or more organizations and have some role.

In this paper, we define "business process" as *a collection of actions performed by particular users or organizations for a purpose*. This definition has a high affinity with the definition of "activity" in the process mining manifesto [15]. By running it with the input information described above, "one purpose" corresponds to the case ID in the system log, and "particular users or organizations" corresponds to the user ID and organizational information. We use this property to recover highly abstracted business processes.

IV. The Proposed Method

A. Overview

Figure 2 provides an overview of the proposed method. The proposed method consists of four steps. The first step is the role model recovery step that recovers the role structure of actions from input information, system logs, and organizational information. The second step is the main actions' selection step that selects the most characteristic actions from those belonging to each activity. The next step is the abstract activity recovery step that recovers abstract activities named the main actions. The last step is the flow drawing step that visualizes the recovered business process using a generic drawing tool. In the following sections, we introduce some definitions and provide the details of each step.

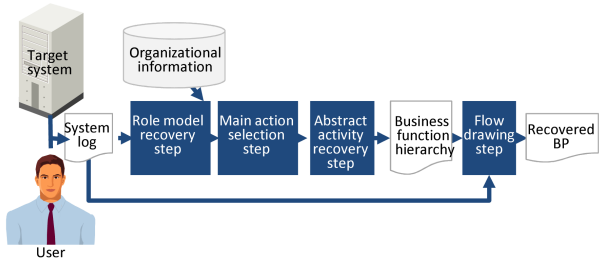

Fig. 2. Overview of the proposed method.

B. Basic Definition

We define system log E as an ordered set of log entries e. Log entry $e_i \in E$ corresponds to a record of a user operation on the target system and defined as follows:

$$e = \langle e.u \in U,\ e.t \in T,\ e.a \in A,\ e.w \in W \rangle \tag{1}$$

where $U = \{u_1,\ u_2,\ u_3,\ \cdots\}$ is a set of *users* and $T = \{t_1,\ t_2,\ t_3,\ \cdots\}$ is a set of *time stumps* when user operations are executed on the system. $A = \{a_1,\ a_2,\ a_3,\ \cdots\}$ is a set of *actions* corresponding to the type of user operations, and $W = \{w_1,\ w_2,\ w_3,\ \cdots\}$ is a set of *cases* (a.k.a work items).

To gather log entries for a case, we can recover a history of user actions for it. We define the history as process instances $p_i = \{e|e.w = w_i,\ e \in E\}$.

The activity $act \in ACT$, which is the key element in the recovered process model, is defined as a set of actions. Our definition allows us to use the arbitrary combination of actions as an activity ($ACT \subset 2^A$).

C. Step1: Role Model Recovery Step

The outline of this step is as follows:

(1) Extract process instance from system logs and identify blocks based on the switching points of users
(2) Calculate $weight(a_i \Rightarrow a_j$ for all relations R between actions and filter out relations $a_i R a_j$ where $weight(a_i \Rightarrow a_J) \leq MIN_{weight}$. Then extract connected actions with relation R as *roles*.
(3) Group roles based on organizational information.

In the rest of this section, we describe the details of each step with examples.

1) Identifying Blocks: In the first step of recovering the structure of a business process, we find candidates of activities called "block" from process instances. In this paper, we focus on the user role that users perform for a business artifact. Usually, a business artifact is related to some user roles. We think that an activity, which is the main element of the business process, can be recovered through the actions of each user role.

Block $b_{i,j} \in B$ is defined as the subset of a process instance related to a particular user; $b_{i,j} = \{e|e.u = u_j, e \in p_i\}$. Based on this definition, all blocks B can be defined as $B = \{b_{i,j}|\forall w_i \in W,\ \forall u_j \in U\}$. Note that a block represents a series of actions by a user for a case. A user tends to perform the same actions for cases as a part of the workflow. In the large business application system described in the last section, some users have the same role in the workflow to

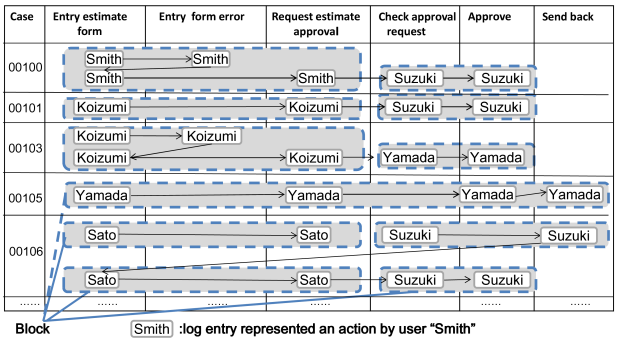

Fig. 3. Example of process instances, block, and role switching points.

perform actions in parallel. For instance, actions in $b_{user1,case1}$ and $b_{user2,case2}$ can be identical when they have the same role in the workflow. However, actions in $b_{user2,case2}$ and $b_{user2,case3}$ might be different, since user2 performs different actions for case2 and case3. If a user has multiple roles, a block for the user contains several activities.

Figure 3 shows an example of process instances and blocks for a sample system log in Fig. 1. In this figure, the vertical lane shows process instances, and horizontal lane shows actions. The rounded rectangles with a broken line represent blocks. In this case, $u_{Koizumi}$ performed different actions in c_{00101}, c_{00104} and u_{Yamada} functioned in two roles in case c_{00105}.

2) Extraction of Roles: To cope with variations in actions in a block, we compute the weighted relation among actions to identify roles. In this paper, we use *lift* as weight. $lift$ is used to measure the distance between two events in association rule mining. We define the lift value from action a_i to action a_j as follows, where $aBlocks(a)$ is a function to obtain a set of blocks that have action a, i.e., $aBlocks(a) = \{b \in B | \exists e \in b,\ e.a = a\}$:

$$lift(a_i \Rightarrow a_j) = \frac{|aBlocks(a_i) \cap aBlocks(a_j)| \cdot |B|}{|aBlocks(a_i)| \cdot |aBlocks(a_j)|} \quad (2)$$

We pick connected actions related by high lift values for roles, i.e., a role r is a set of actions which hold $\forall a_i \exists a_j \in r.\ lift(a_i \Rightarrow a_j) > MIN_{weight}$. It is based on the property whereby the actions of the same role are prone to be performed by same user at the same time. The frequencies of actions in the log have a wide range; hence, we selected a lift value to reduce the influence of high-frequency actions in clustering. The algorithm picks a group of actions related by lift value above the threshold for the role.

Based on the lift value, we can eliminate weak relations, such as the one between actions performed by a user with multiple roles and find the roles including variations in the routine work. For instance, the process instance of c_{00105} in Fig. 3 has four actions in a block. However, lift value correspondence between "Request Estimate Approval" and "Check Approval Request" is low because these actions were performed by different users in other cases. On the other hand, lift value correspondence between "Entry Estimate Form" and "Request Estimate Approval" is high. Thus, "Request Estimate Approval" and "Check Approval Request" belong to different roles, whereas "Entry Estimate Form" and "Request Estimate Approval" are grouped in a role.

Organization	Role	Action
The Sales Division	Role1	Entry estimate form
		Entry form error
		Request estimate approval
	Role2	Check approval request
		Approve
		Send Back
Inventory Management Division	Role3	Receive estimate
		Request inventory
		Receive request
		Reserve inventory

Fig. 4. Recovered role model of Fig. 3.

3) Reflecting Organizational Information: To understand the big picture in the process modeling of a large-scale system, we need higher-level abstraction. Hence, our method uses not only user information, but also organizational information, such as sections, divisions or departments, to recover high level roles. Each user who performs an action, has organization. In this step, we group roles into highly abstracted ones by assigning to them the organization that most frequently appears in actions in a given the role. That is to say, we recover relations between an organization and an action from relations between the organization and the userID by selecting the most frequent organization to which the users acting given actions belong. Figure 4 shows result of this step for the example from Fig. 3.

D. Step2: Main Action Selection Step

To recover the business process from the model of roles, we find main actions that are characteristic actions. Usually, users of business applications start their operations from general actions like "check request," or "form error." They then perform various actions, and finish it by the characteristic actions like "Approve" or "request approval."

We focus on actions performed at the end of a role frequently. We define a function $fRate(a) = |fBlocks(a)|/|aBlocks(a)|$ to calculate the rate of the *final appearing blocks* defined as set of blocks where the focused action appears at the end. $fBlocks(a)$ is a function to choose final appearing blocks for an action a, i.e., $fBlocks(a) = \{b \in aBlocks(a) | \exists e_0 \in b, e_0.a = a, \forall e_k \in b\ e_0.t \geq e_k.t\}$

In this paper, we select the main actions of a given role r with a function $mainAct(r) = \{a \in r | fRate(a) \geq MIN_{final}\}$.

E. Step3: Abstract Activity Recovery Step

In the last section, we described the algorithm to select the main actions. To recover the abstracted activity, our method gathers blocks containing each main action, and selects actions more frequent than the threshold value. We define these groups of actions as abstracted activities named the main action. In some cases, one action belonged to more than one abstract activities or no abstract activities.

F. Step4: Flow Drawing Step

The last step of the method involves drawing a flow diagram from the recovered information. It generates a frame with

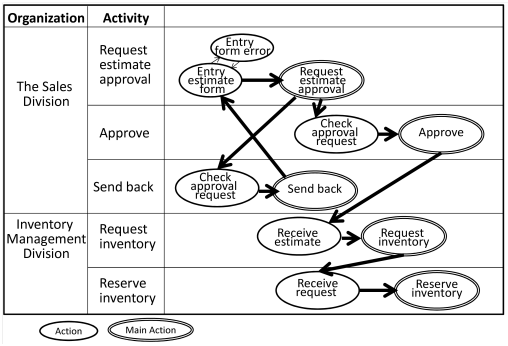

Fig. 5. Business process diagram (Operation level).

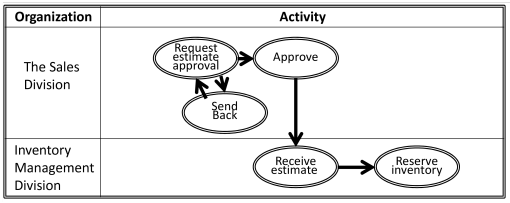

Fig. 6. Business process diagram (Activity level).

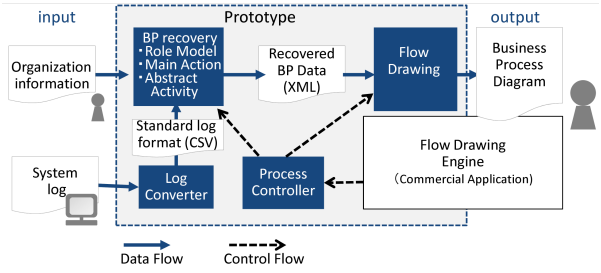

Fig. 7. Outline of our prototype system.

TABLE I
EVALUATION OF RECOVERED ABSTRACT ACTIVITIES

Business Process	Correct transitions	Incorrect transitions	Precision
BP.A	15	0	100%
BP.B	15	2	88%
BP.C	6	3	67%
BP.D	7	2	78%
BP.E	6	2	75%
Total	49	9	84%

the organization and the abstracted activities, and connects activities linked by the process instance sequence (Fig. 5). Furthermore, it folds actions in abstract activities to generate a highly abstracted business process by looking up the main actions (Fig. 6).

V. Implementation

Figure 7 illustrates a prototype tool we developed. The tool consisted of four steps described in the last section, and was called the "log converter". It converts real system log into a standard log format like that shown in Fig. 1, and the "process controller", controls each step and activates the steps. We implemented the "flow drawing step" with a macro-program of a commercial flow-drawing application.

VI. Case Study

A. Target System

We selected an business application system that supported preliminary petitions, adjustments and approval for business trips. The system consisted of five subsystems related to five business processes (BP.A, BP.B, ... , BP.E). The roles of the users were: employee, who wanted to go on a business trip; supervisor, who approved the employee's petition for the business trip; an accountant, who considered the budget. These users used the system via a Web browser.

We obtained an actual system log (personal data was masked) and organizational information concerning users for this case study. The system log was 6.22M bytes, and included actual usage data for three months. The log consisted of 18,064 log entries related to 3,969 users, 59 actions, and 9,946 cases.

B. Evaluation Method

To evaluate the validity of the recovered abstract activities, a senior developer who knew the system well, checked the validity of the abstraction and the transitions between abstract activities. We also evaluated the performance of our main action selection method. We choose BP.A as a target business processes and defined the conventional method, which selected the most frequent action in an activity, as the base line. We compared our proposed method with the conventional method in terms of precision and recall of selecting main actions.

C. Evaluation Result

1) Validity of Recovered Abstract Activities: We recovered an abstract activity of the target system using the proposed method. We asked a senor developer of the target system to evaluate the abstract activity flow using two points of view, the qualitative validity of abstraction and the quantitative validity of transitions between abstract activities. From the first point of view, the developer checked recovered abstract activities to look for missing or unnecessary activities, and concluded that it was satisfactory. Table I shows the result from the second point of view. The precision of our method was 67%-100% (average 84%). Incorrect transitions consisted of related common actions over MIN_{lift}.

Figure 8 and Figure 9 are screen shots of the recovered flow of actions and the abstracted activity flow of BP.A, respectively. Our method successively recovered activities and reduced the size of elements in the flow diagram: 59 actions into seven abstract activities in this case. For example, the actions "display detail form," "save form," "update hotel information," "request domestic trip approval," and so on, abstracted the activity "request domestic trip approval."

2) Validity of Recovered Main Actions: Table II shows the evaluation of the selection the main action in BP.A. It compares our method (*fRate*) and the simple method, the selection of the most frequent action(Frequency). Our method with Min_{final} = 75% yielded the best precision and recall scores. When Min_{final} was under 75%, precision was low because intermediate actions, like "delete item" or "search case," were selected as main actions. When Min_{final} was 90%,

TABLE II
EVALUATION OF MAIN ACTION SELECTION

	fRate				Frequency			
	30%〜	50%〜	75%〜	90%〜	30%〜	50%〜	75%〜	90%〜
Main Actions	10	9	7	4	5	4	1	1
Correct answer	7	7	7	4	3	2	1	1
Precision	70%	78%	100%	100%	60%	50%	100%	100%
Recall	100%	100%	100%	57%	43%	29%	14%	14%

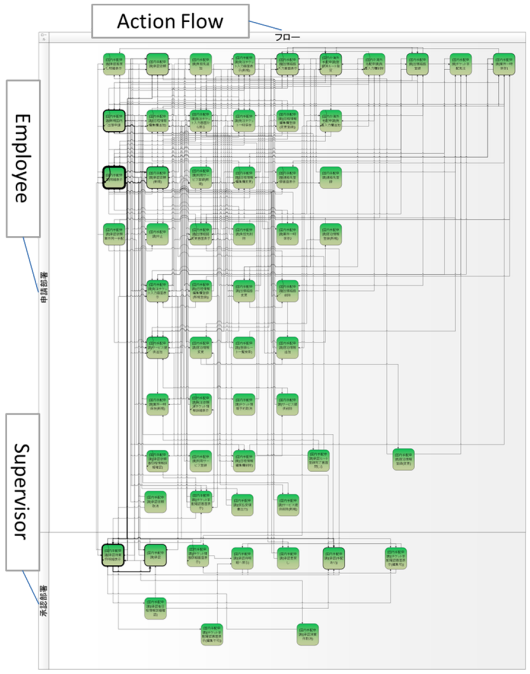

Fig. 8. Recovered business process (before abstraction).

recall was low because important actions were eliminated. In all cases, the proposed method outperformed the baseline method.

D. Discussion

In this case study, the proposed method was effective in recovering abstract activities using system log. By separating process instances into blocks and grouping them according to lift values, we recovered valid abstract activities. Furthermore, by using organizational information, we recovered higher-level abstracted activities. On the contrary, a few frequent actions led to incorrect results in the evaluation of transitions between abstract activities. There is hence room for improvement.

And by focusing on the $fRate$, the main action of abstracted activity is recovered with high precision and recall. We should do more case studies to realize the best threshold of $fRate$. For good result, our method requires an entire lot of log entries. Most of business applications have large size logs.

VII. CONCLUSION

To understand the specifications of legacy business systems, we proposed in this study a method to recover business processes from system log and organizational information. Our method is intended for business processes performed by a plurality of users. It focuses on change points of roles,

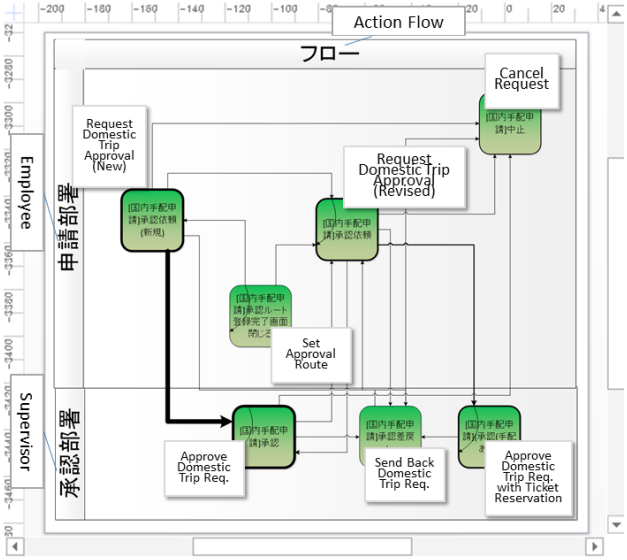

Fig. 9. Recorvered business process (after abstraction).

categorizes actions into abstracted activities, and selects the main action of the each activity.

In our case study involving a real system, we recovered seven abstracted activities from 57 actions, and selected the main action of each activity with high precision and recall.

REFERENCES

[1] G. C. Murphy, D. Notkin, and K. Sullivan, "Software reflection models: Bridging the gap between source and high-level models," in *Proc. FSE'95*, pp. 18–28.

[2] A. Rountev and B. H. Connell, "Object naming analysis for reverse-engineered sequence," in *Proc. ICSE'05*, pp. 254–263.

[3] L. A. P. Rabelo, A. F. do Prado, W. L. de Souza, and L. F. Pires, "An approach to business process recovery from source code," in *Proc. ITNG'15*, pp. 361–366.

[4] A. Ghose, G. Koliadis, and A. Chueng., "Process discovery from model and text artifacts," in *Proc. SOPOSE'07*, pp. 167–174.

[5] B. Paradauskas and A. Laurikaitis., "Business knowledge extraction from legacy information systems," *J. Info. Tech. and Control*, vol. 35, no. 3, pp. 214–221, 2006.

[6] Y. Zou, T. C. Lau, K. Knotogiannis, T. Tong, and R. MeKegney, "Model driven business process recovery," in *Proc. WCRE'04*, pp. 224–233.

[7] C. D. Francescomarino, A. Marchetto, and P. Tonella, "Cluster-based modularization of processes recovered from Web," *J. Softw.: Evol. and Proc.*, vol. 25, no. 2, pp. 113–138, 2010.

[8] R. Pérez-Castillo, "Marble: Modernization approach for recovering business processes from legacy information systems," in *Proc. ICSM'13*, pp. 671–676.

[9] M. Leemans and W. van der Aalst, "Process mining in software systems: Discovering real-life business transactions and process models from distributed systems," in *Proc. MODELS'16*, pp. 44–53.

[10] W. van der Aalst, *Process Mining - Data Science in Action*. Springer, 2016.

[11] J. D. Weerdt, S. vanden Broucke, J. Vanthienen, and B. Baesens, "Active trace clustering for improved process discovery," *IEEE TKDE*, vol. 25, no. 12, pp. 2708–2720, 2013.

[12] C. D. Francescomarino, A. Marchetto, and P. Tonella, "Reverse engineering of business processes," in *Proc. CSMR'09*, pp. 139–148.

[13] J. D. Weerdt, M. D. Backer, J. Vanthienen, and B. Baesens, "A critical evaluation study of model-log metrics in process discovery," in *Proc. BPM'10*, pp. 158–169.

[14] ——, "A robust f-measure for evaluating discovered process models," in *Proc. CIDM'11*, pp. 148–155.

[15] W. van der Aalst *et al.*, "Process mining manifest," in *Proc. BPM 2011*, 2012, pp. 169–194.