Process Discovery and Guidance Applications of Manually Generated Logs

Stefan Schönig, Christoph Günther, Stefan Jablonski
University of Bayreuth
Chair of Applied Computer Science IV
Bayreuth, Germany
{stefan.schoenig, christoph.guenther, stefan.jablonski}@uni-bayreuth.de

Abstract - **In this paper, we investigate the problem of the availability of complete process execution event logs in order to offer automatic process model generation (process discovery) possibility by process mining techniques. Therefore, we present the Process Observer project that generates manual logs and guides process participants through process execution. Like this, our project offers the possibility for the automatic generation of process models within organizations, without the availability of any information system. Process participants are encouraged to work with the Process Observer by various process execution support functions, like an auto-suggestion of process data and dynamic recommendations of following processes.**

Keywords - Process Mining, Process Monitoring, Activity Tracking, Guidance through Process Execution

I. INTRODUCTION

Business process management (BPM) is considered an essential strategy to create and maintain competitive advantage by modeling, controlling and monitoring production and development as well as administrative processes [1] [2]. Many organizations adopt a process based approach to manage various operations. BPM starts with a modeling phase, which is very time and cost intensive. It requires deep knowledge of the underlying application and long discussions with the domain experts involved in the processes in order to cover the different peculiarities of the process [3]. Since process modeling is an expensive and cumbersome task, we identify approaches that promise to reduce the modeling effort. One of them is process mining. Process mining utilizes information/knowledge about processes whilst execution. The idea is to extract knowledge from event logs recorded by information systems. Thus, process mining aims at the (semi-)automatic reconstruction of process models using information provided by event logs [4]. The computer-aided creation of process models offers huge potential of saving time. By deriving process models from event logs, the appropriateness of process models can be guaranteed to a certain extent, since they are constructed according to the way the processes have actually been executed. During the last decade, many techniques and algorithms for process mining have been developed and evaluated in different domains [5]. The basis for a successful generation of a process model through process mining is an existing and complete process execution log. This is also the big challenge for a successful application of process mining. First of all, not all processes are executed by information systems, i.e., they are executed "external" to computers. In such cases, there is no event log that represents a process available and process mining cannot be applied. In the case that information systems are already used to execute processes there must be guarantees that these event logs record process execution in such a way that processes can be reconstructed. Besides, these event logs must be analyzable in such a way that appropriate process models can be derived. It is obvious: the quality and availability of event logs determine the applicability of process mining techniques. Our research starts with the assumption that a complete and freely analyzable event log is usually not available. We regard this scenario as the most common one. Thus, one of the major aims of our research is to harvest process execution knowledge. This enables the assembly of a process execution log. This log is built up independently from the existence of information systems that are (at least partly) executing the processes. We developed a special software, the Process Observer (PO), that can be envisioned as a tool that permanently runs on the computers of process participants that asks the process participants “What are you doing right now?”. The participants then have to describe what they are doing. Here, the user does not need any process modeling skills. This is also one very important prerequisite since we assume that just few process participants do show process modeling skills. The recorded data is used by the PO to mine for process models. Of course, this process information can be enriched and complemented by event logs from information systems that are involved in the process execution. Gathering process execution information comes with the cost that process participants have to record what they are doing. Of course, this means additional work for the process participants. Therefore, the PO must offer a stimulus that motivates process participants to work with the PO. This stimulus is put into effect by a recommendation service. The PO continuously analyzes available process log data to guide the process users. This means, it suggests process steps that the user most probably should perform. We have experienced that this feature is especially important for users that are still not too familiar with the application; they are thankful that the PO recommends possible process steps. This dynamic recommendation service becomes more and more reliable the more process instances have been executed under the guidance of the PO. The execution of first instances of a process will therefore not considerably be supported. The effect becomes apparent when a couple of process instances have been executed. At the end of this introduction, we want

 ISBN: 978-1-61208-201-1

to classify the PO. As dimensions for this classification we take the two issues: attaining a process model and executing a process model. We already discussed the two principal approaches to attain a process model. They will be assessed with respect to the amount of effort a process participant has to or is able to invest. The first approach to attain a process model is to create it within a process modeling project. This task is very costly; it usually cannot be performed by process participants but requires process modeling experts. They identify the process through interviews with the domain experts and need to get a good overview over all possible process peculiarities to guarantee the completeness of the process model. Process models can also be attained by the application of process mining techniques. This approach is cheap since only little work from process modelers is required. However, it depends on the existence of event logs representing the execution of processes. These two approaches depict two extreme landmarks: on the one hand processes can be performed within information systems. On the other hand, information systems could not be involved at all. The PO bridges the contrary approaches of process execution und thus combines their benefits. It is connectable to process execution systems and can leverage them; also it provides execution support for "external" process execution.

In Section II, we will give an overview over related works. In Section III we will explain our concepts and the general approach. Furthermore, concrete implementation techniques will be shown in Section IV. Section V describes the influence of the PO on the current process lifecycle. In Section VI we will finally conclude and give an outlook on further research issues and applications.

II. RELATED WORK

The idea of automating process discovery through event-data analysis was first introduced by Cook and Wolf in the context of software engineering processes [6]. In the following years, Van der Aalst et al. developed further techniques and applied them in the context of workflow management under the term process mining [5]. Generally, the goal of process mining is to extract information about processes from event logs of information systems [7]. There are already several algorithms and even complete tools, like the ProM Framework [8], that aim at generating process models automatically. During the last decade, several algorithms have been developed, focusing different perspectives of process execution data. Van der Aalst et al. give a detailed introduction to the topic process mining and a recapitulation of research achievements in [5] and [7]. For the first prototype of the PO, we use the alpha-algorithm of [3]. However, for our future research activity we consider algorithms like the HeuristicsMiner [9] appropriate, because they are able to deal with noisy logs, i.e., incorrectly or incomplete logged information. Process mining algorithms rely on complete event logs from information systems. In the case of an incomplete log or even the unavailability of an information system, events can alternatively be recorded by manual activity tracking respectively task management methods. There are several approaches for activity tracking to generate weakly-structured process models by capturing data on personal task management [10] [11]. However, these approaches lack the use of process mining techniques during and after process run-time. In contrast to that we explicitly try to encourage user contribution to an evolving process model by using process mining methods. In order to discover identical processes between different data storages, we suggest using basic automatic ontology matching algorithms [12]. Process mining is considered as a part of Business Process Management (BPM). BPM relies on a life-cycle where different phases of the process are focused. The traditional approach consists of the following phases: process modeling, implementation, execution and evaluation, started by the modeling step. Despite the successful development and evaluation of the process mining algorithms named above, process mining is ranked among the process evaluation phase [1]. Consider, for example, Enterprise Resource Planning (ERP) systems such as SAP, OpenERP, Oracle, Customer Relationship Management (CRM) software, etc. These systems require a designed process model before they go into service [3]. In these situations, process mining could only be used for process rediscovery and not for real process discovery. Therefore, we aim at assigning process mining to the discovery phase by recording the complete process data covering all aspects of the perspective-oriented process modeling (POPM). In order to get a general idea about POPM perspectives, we recommend [13] and [14].

III. GENERATION OF PROCESS EXECUTION LOGS AND GUIDANCE THROUGH PROCESS EXECUTION

Process mining techniques allow for automatically constructing process models. The algorithms are analyzing a process execution log file, in the following referred to as (process) log; this log is usually generated by information systems (IS). However, there are processes that are not executed by information systems. This is an observation that is very important for the classification of our research. Thus, in order to define the application area of our project we have to introduce three different types of process execution support, classified upon the degree of logging and execution support (Fig. 1):

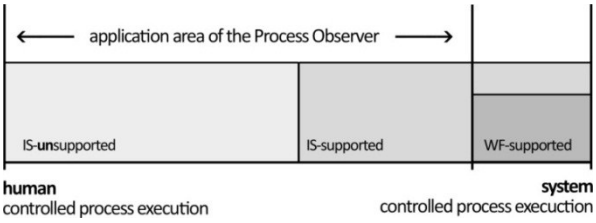

Figure 1. Application area of the Process Observer project

- *IS-unsupported*: Here, processes are executed without the support of any information system. Thus, there is no log for these processes. Furthermore, these processes are also not supported during execution. For example, there is no information system that guides a user through the process.

- *IS-supported*: Here, processes are executed by an information system. Processes of this type are (possibly) logged. However, the information system is not directly guiding users through the process. The user has to find his way through the information system by himself.
- *WF-supported*: Here, processes are executed by Workflow management systems (WFMS). WFMS build a subset of IS. Typically, they maintain a process log. Additionally, the process participants are guided through process execution with concrete recommendations of how to continue process execution (work list) [15].

The basis for the successful generation of process models through process mining is an existing and complete log. Thus, WF-supported processes are a great source for process mining. Nevertheless, the existence of a process log is the main prerequisite and also the major drawback for a successful application of process mining. Since we assume that in many applications, WF-supported processes will not be encountered the PO turns its attention to IS-supported and IS-unsupported processes (Fig. 1). In order to log IS-unsupported processes, we extend process execution by manual logging. We define the term manual logging as the user action of entering process execution data (e.g., process IDs, documents, and services) as well as of marking process execution events, among other things process start and completion. The action of manual logging is implemented by the PO Logging Client. Finally, our goal is to provide manual logging in such cases when processes are neither IS-supported nor WF-supported. The final aim is then to be able to apply process mining.

A. Aims of the Process Observer

The challenge of the PO is to provide a broader basis for process mining by implying IS-unsupported processes in logs. Therefore, the PO project aims at the adoption and generation of manual logs. The generated manual logs open the opportunity for the automatic generation of process models by process mining techniques even for applications that do not involve information systems. As manual logging is performed by process participants, it means additional work for them. Therefore, the PO must offer a stimulus that motivates process participants to support manual logging. Since the PO is particularly of interest for IS-unsupported and IS-supported processes, it offers a stimulus with respect to process execution guidance (this is what these two kinds of processes are lacking). The PO offers recommendations about how to continue a process execution and offers auto-suggest support. This kind of guidance during process execution is typically exclusively offered by WFMS.

B. Generation of Manual Logs

From now on, we generally assume that a complete and freely analyzable log is not available, i.e., we are focusing on IS-(un)supported processes. We regard this scenario as the most common one and it needs to be supported to apply process mining.

1) Manual Logging:

The generation of a manual log is initiated by the PO Logging Client. Process participants record what they are currently doing, i.e., they provide information about the process they are currently performing. It is very important that users do not need any process modeling skills to record this information.

An important issue is to determine what data the process participants should record. We recommend to record data based upon the different aspects of perspective oriented process modeling (POPM). We have experienced that most users are very familiar with the approach of describing process in the POPM method. Process participants have to enter data according to the following perspectives:

- *Functional* perspective: name of the current process step, the name of the corresponding superordinate process (if available)
- *Data* perspective: data, i.e., documents or generally information that was used by the current process step as well as the data or documents that were produced
- *Operational* perspective: tools, applications or services that were used during the execution of the currently executed process step
- *Organizational* perspective: information about the process executor (typically, this is that person that is logged into the PO Logging Client), the personal information is enriched by group and role memberships

Besides, process participants have to trace process execution: he has to declare that process execution starts, ends or is aborted.

2) Merging Logs:

The application of the PO Logging Client finally results in the generation of a manual log. In the case, that an information system is applied, there might also be an automatic log available. We harness this situation by combining the manual log with the automatic log. Doing this, missing process information of one of the logs can be completed by the other log. In order to be able to combine the two logs, conformance between the recorded data of both logs must be achieved. Therefore, we suggest a component for merging the logs, i.e., locating (matching) and unifying processes that were recorded in the manual log as well as in the automatic log. This results in one consistent log that contains the execution data of IS-unsupported as well as IS-supported processes.

C. Guidance through process execution

According to our classification in Fig. 1, many process executions are not assisted by a guidance component, i.e., the participants must decide for themselves which process step they want to perform next. Only WF-supported processes do provide this feature. In this subsection, we will show how the PO offers such guidance. It consists of two sub-features: dynamic recommendations and auto-suggest function.

 ISBN: 978-1-61208-201-1

1) Dynamic Recommendations:

Dynamic recommendations are generated in the following way: After the completion of a process step, the PO immediately starts a process mining algorithm analyzing available log data. It then tries to classify this current process execution into former process executions. If it is successful, the PO can recommend the execution of a subsequent process step according to the processes that have been executed formerly. This recommendation service becomes more and more reliable the more process instances have been executed under the guidance of the PO. When only a few or even none processes of this type have been executed so far, no recommendations can be made for the particular process. Especially when only a few process instances have been performed so far, the recommendation can be inconsistent. Then, process participants can ignore this recommendation. In order to know about the quality of the recommendation, the number of process instances the recommendation is based upon is displayed in the user interface.

Example: A process participant just completed a process step A. This step has already been completed and recorded 10 times before by other agents. On the one hand, step B was executed 7 times after step A; on the other hand, step C was executed 3 times after step A. The PO now starts process mining and generates a process model that contains the information that process A shows two subsequent processes B and C. Furthermore, the tool takes into account that step B occurred 7 times and step C occurred 3 times after step A in the log. Thus, a dynamic recommendation is shown to the user suggesting to continue with step B (70%) or step C (30%).

2) Auto Suggest Function:

The second aspect of guidance during process execution is provided by an auto-suggest function. This function helps the process participant to enter required information. The PO compares previously recorded process names, data, tool names, etc. with the currently entered term and auto-suggests terms. This function supports two issues: first, the user might nicely be supported through information provision; secondly, by suggesting already used terms, the probability of having to deal with too many aliases in the system is diminished to a certain extent.

Example: Agent 1 is executing a process "Drinking Coffee". Agent 1 starts the process by recording the process name, i.e., Agent 1 enters "Drinking Coffee". The agent starts and completes the process. The process gets a unique identifier and is recorded in the log. Later, Agent 2 also wants to drink coffee and executes this process with support of the PO. He starts by typing "Coffee" instead of "Drinking" in the process name row. This would easily result in the recording of a process name like “Coffee Drinking” or just "Coffee". So, aliases are produced without even recognizing. However, in this case an auto suggestion will appear, recommending to choose the process "Drinking Coffee". Agent 2 happily chooses the suggested process and thus ensures homogenous naming of the process step.

3) Visualization and manual mapping of processes:

Example: If the example from the former sub-section would occur as described, this would be ideal. However, in many cases same processes will be referenced by different aliases and thus stay unrecognized by the PO. In order to handle problems like this, the PO offers an administration interface, which allows process administrators to visualize recorded processes. Administrators can start process mining algorithms and thus generate process models visualizing observed processes. Doing this, different aliases of processes can be discovered. However, this must be done manually by the administrator. In order to map different aliases of the same process, the PO administration interface offers a mapping panel. This mapping can be declared valid for multiple processes (Fig. 2). After defining a mapping between processes, a repeated execution of process mining results in the visualization of the amended process model.

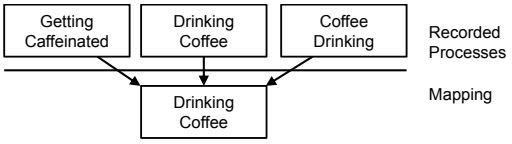

Figure 2. Sample mapping of recorded processes

D. Usage scenarios for the Process Observer

As a conclusion, we will give a short description of three different application scenarios of the PO.

1) Use Case 1 – Generation of manual logs:

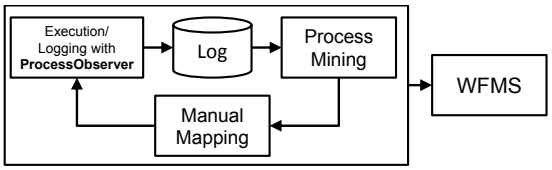

Figure 3. First use case – generation of manual logs

The first use case comprises the generation of a manual log (Fig. 3) without an information system being available. The participating agents are executing the corresponding processes under the guidance of the PO. The manual log is finally analyzed by process mining algorithms. The resulting process models can be fed into a WFMS if wanted and if available. Thus, processes can afterwards be executed by a WFMS.

2) Use Case 2 – Merging of logs:

The second use case comprises the application of the PO in parallel to an information system (Fig. 4). After the generation of a manual log, we have to merge the automatic and the manual log.

 ISBN: 978-1-61208-201-1

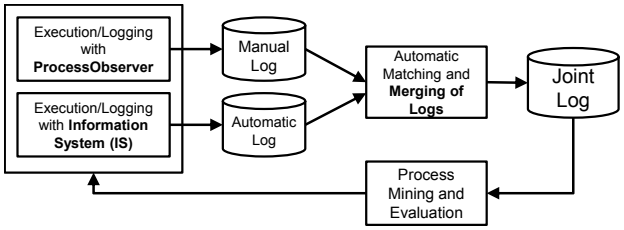

Figure 4. Second use case – merging of logs

The intention is to complete the log information mutually. Identical processes are merged to one single process. Process Mining is finally applied to the joint log. Identified processes can be fed back into information systems.

3) Use Case 3 – Running WFMS:

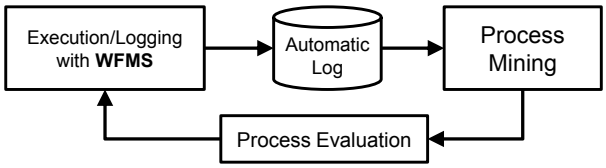

Figure 5. Third use case – running WFMS

The third use case assumes a fully-fledged WFMS running (Fig. 5). Here, manually logging is not necessary anymore because the WFMS includes all the processes being executed. It is important to define a threshold, when process management can shift from case 2 to case 3. Therefore, we define a value *matching_count* (1) as the number of matched processes from the manual log and the automatic log divided by the complete number of processes recorded in the manual log. The procedure of calculating this value is the following: the algorithm runs through both logs. It compares each process of the manual log with the processes of the automatic log. If an ontology matching algorithm identified two processes as equal, the numerator *#matched_processes* will be increased by 1. After finishing traversing both log files, the resulting value of *#matched_processes* is divided by the total number of recorded processes within the manual log.

$$matching\_count = \frac{\#matched\_processes}{\#recorded\ processes\ with\ Process\ Observer} \quad (1)$$

Like this, the calculated value reflects how many processes are already executed with support of the WFMS. Generally, an organization finally tries to execute all processes under the guidance of the WFMS, but the preferred value of *matching-count* can also alternatively be defined by the management. For a special organization a *matching-count* value of 0.9 may be enough. This means, 90% of the executed processes are implemented and supported by the WFMS. Like this, the right time of the application end of the PO can be declared by continuously calculating the *matching_count* (1) value.

IV. ARCHITECTURE AND IMPLEMENTATION

In this section, we will describe the architecture and implementation of the PO. In the first part, we will show implementation details of the PO Logging Client. After that, process mining implementation and data structures will be explained. Furthermore, we present the administration and mapping components.

A. Process Observer Logging Client

The core of the PO is constituted by the PO Logging Client. We decided to choose a web based implementation of the logging interface. This guarantees a great coverage of application scenarios, i.e., the PO can be used in almost all applications. If the users are working in a "normal" office, the PO can run on a stationary PC or notebook, if users are working "in the field", the PO could as well run on a mobile device (e.g., smartphone). For our prototype we chose an implementation based on Microsoft ASP.NET 4.0 and the MS SQL Server 2008 database, but surely any equally equipped database and server technology would be suitable. The core of the web application that implements the PO Logging Client is located on a web server connected to a database. Users have to identify themselves by logging in with their username and password. Users can be assigned to one or more organizational roles. Hence, recommendations and suggestions can be personalized to the users' roles. When users enter process names they want to log, these text strings are immediately sent to the PO to test for similar process names. The names of all processes containing a similar string are sent back to the client as a generic list. This list is finally displayed to the user as an auto suggestion list (Fig. 6). The user can select a process from this list. If none of the suggested processes is appropriate, the input process name is added as a new process. Accordingly, all other process data are captured (e.g., superordinate process, current process instance, used and produced data/documents and supporting tools). Finally, the user starts the process.

Figure 6. Example of auto suggestion list

B. Implementation of process mining, data structures and dynamic recommendations

As already described in Section III, the PO offers dynamic recommendations of how to continue after finishing a process step. Therefore, a process mining algorithm is executed after each process step. In our prototype we use the alpha algorithm of [3] in order to analyze the available logging information. The algorithm analyzes the log and builds up a dependency graph. Therefore, we used the graph data structure QuickGraph of

[16]. For implementation details concerning the alpha algorithm we refer to [3]. The logged execution information results in process models represented as graphs. A node is an instance of a class "Process" containing fields for process name, the executing originator role, used and produced data items as well as supporting tool items. Furthermore, the class contains two fields for the pre- and post-connectors which represent the semantic connection to previous and following processes. This information is also provided by the alpha algorithm. Once a process model has been generated as a graph, the PO can use it in order to display recommendations after a user has finished a process step. Therefore, the recently completed process is searched within the process model, i.e., the graph is traversed until the current process ID is identical to the recently completed one. After that, all available edges of this node are examined and their occurrence is counted. Like this, we generate a list, containing the processes that were executed after the recently completed one. Thus, a popup is displayed, giving the user the possibility to choose the following process step.

C. Administration interface

Additionally, the PO offers an administration interface that allows process administrators to visualize recorded processes as well as defining mappings between logged processes as described in Section III. The application consists of two panels, one for process model selection and visualization and the other one for defining mappings between processes. One could easily imagine additional applications, like agent-role assignments or dataflow applications. Those are planned for future versions.

1) Process visualization:

In order to visualize the generated process model we use basic graph visualization frameworks. In our prototype we used the Graph# framework [17] to display the QuickGraph data structures. The visualization procedure is started by examining the recorded event log for contained composite processes. A process is recognized as composite, if it was chosen as a superordinate process by a process participant during the logging phase of a process with the PO. The names of the composite processes are loaded in a tree view. The user selects a composite process that should be displayed from the tree view. The tree view shows the underlying process hierarchy. Processes that are contained within another one can be displayed by extending a process entry. After the selection of an entry, all event log information concerning the selected process is fetched from the database. After that, the alpha algorithm is applied to the resulting event log data. As stated before, the algorithm generates a dependency graph. This graph is finally assigned to the Graph# framework and displayed to the user. Here, the user has various possibilities to scroll within the visualization or to open the model of underlying composite processes by selecting the corresponding process nodes.

2) Mapping definition panel:

Furthermore, the administration interface offers a separate panel to define mappings between logged processes. Therefore, the database provides a separate mapping table with three columns: “superordinate process”, i.e., the super process within the mapping is valid, “target process”, i.e., the process on which another one is mapped and finally “mapped process”, i.e., the process which is mapped. Considering this data model, the mapping panel consists of three columns, too. They appear after the first things first principal. In the first list, the user selects the superordinate process within the mapping should be valid. After this selection, the target process list appears. The list is initialized with all processes occurring within the chosen superordinate process. Like this, the user can choose the target process for the defined mapping. Last but not least, the last list, i.e., a checkbox list, appears. It is again initialized with all processes of the corresponding super process. Here, the user checks all the corresponding boxes of the processes he would like to map on the target process chosen before. Finally, the mapping is applied to the database.

V. CHANGES WITHIN THE PROCESS LIFECYCLE THROUGH THE PROCESS OBSERVER

In this section, we will describe the impact of the PO on different phases in the process lifecycle. As already mentioned, the previous process lifecycle [1] consists of an initial modeling phase that is very time consuming. In this lifecycle, process mining is only used for the evaluation of the process being executed with support of a WFMS. As any WFMS needs at least one predefined process model in order to be operable [3], there is no possibility to support the intense process modeling phase with the automatic process discovery possibilities of process mining. The development of the PO offers the possibility to change this situation. With support of the PO, the lifecycle can be rearranged in the following way (Fig. 7). The initial step consists of process execution (as usual) accompanied by manual logging, i.e., the generation of a manual log, with the PO. This phase is followed by a process mining step. Afterwards, the results of process mining possibly have to be reworked in a process remodeling phase. The benefit of the application of the PO consists of the time saving between the previous process modeling phase and the less time consuming remodeling phase.

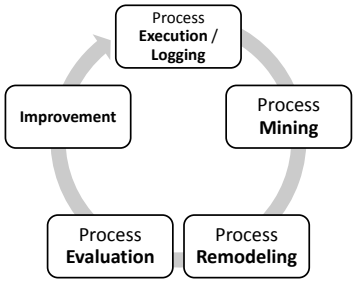

Figure 7. Adapted process lifecycle through the application of the PO

 ISBN: 978-1-61208-201-1

The previous modeling phase, i.e., the project of process discovery and process definition, had to be operated completely manual. The process management team had to do several interviews with agents, live observations of processes and the tracking of documents, for example. In contrast to that, process discovery with the PO is generally more automatable. Merely reworking effort is required in order to annihilate possibly occurring exceptions or execution errors. Based on the results of these first three steps, business processes can be evaluated and finally optimized.

VI. CONCLUSION AND OUTLOOK

In this paper, we discussed the problem of the availability of complete process execution event logs in order to offer automatic process model generation possibility by process mining techniques. Therefore, we presented the Process Observer (PO) project that generates manual logs and guides process participants through process execution. Like this, our project offers the possibility for the automatic generation of process models within organizations, without the availability of any information system. Process participants are encouraged to work with the PO by various process execution support functions, like the auto-suggestion of process data and dynamic recommendations of following processes. This kind of guidance during process execution is typically exclusively offered by WFMS. Our future research activity in the field of the PO will start with the development of matching methods in order to match and merge identical processes. We will also implement a module to transfer the recorded process data into the new the eXtensible Event Stream (XES) format [18]. Furthermore, we will face the problem of recording and logging processes in different granularities. This research faces one of the great challenges of process mining declared during the meeting of the IEEE Task force on process mining at the BPM conference in 2011. In order to deal with execution exceptions and wrongly logged processes, we will implement a heuristic process mining algorithm [9]. Like this, some of the manual mapping activity will be obsolete. Additionally, the control-flow mining algorithm should be featured by decision mining [4] in order to enrich the process models with decision information based upon data extensions. Furthermore, we are developing a new process discovery approach based upon explicit semantic definitions. Finally, we are looking forward to an extensive application of the PO in an organization, accompanied by a detailed documentation of the practice.

REFERENCES

[1] Zur Muehlen, M. and Ho D.: "Risk management in the BPM lifecycle". In: Business Process Management Workshops, LNCS, vol. 3812, 2006, pp. 454-466.

[2] Zairi, M.: "Business process management: a boundaryless approach to modern competitiveness". In: Business Process Management Journal, vol. 3, 1997, pp. 64-80.

[3] Van der Aalst, W., Weijters, T., and Maruster, L.: "Workflow mining: Discovering process models from event logs". In: IEEE Transactions on Knowledge and Data Engineering, vol.16 (9), 2004, pp. 1128-1142.

[4] Rozinat, A. and Van der Aalst, W.: "Decision mining in business processes". In: BPM Center Report BPM-06-10, BPMcenter.org, 2006.

[5] Van der Aalst, W. and Weijters, A.: "Process mining: a research agenda". In: Computers in Industry, vol. 53 (3), 2004, pp. 231-244.

[6] Cook, J.E. and Wolf A.L.: "Automating process discovery through event-data analysis". In: 17th International Conference on Software Engineering, Apr. 1995, Seattle, USA.

[7] Van der Aalst, W., Reijers,H., Weijters, A., Van Dongen, B., De Medeiros, A., Songa, M., and Verbeek, H.: "Business process mining: An industrial application". In: Information Systems, vol. 32 (5), 2007, pp. 713-732.

[8] Van Dongen, D., De Medeiros, A., Verbeek, H., Weijters, A., and Van der Aalst, W.: "The ProM framework: A new era in process mining tool support". In: Applications and Theory of Petri Nets, LNCS, vol. 3536, 2005, pp. 444-454.

[9] Weijters, A., Van der Aalst, W., and De Medeiros, A.: "Process mining with the heuristics miner-algorithm". Department of Technology, Eindhoven University of Technology, 2006, Eindhoven, The Netherlands.

[10] Witschel, H., Hu, B., Riss, U., Thönssen, B., Brun, R., Martin, A., and Hinkelmann, K.: "A collaborative approach to maturing process-related knowledge". In: Business Process Management, LNCS, vol. 6336, 2010, pp. 343-358.

[11] Stoitsev, T., Scheidl, S., Flentge, F., Mühlhäuser, M.: "From Personal Task Management to End-User Driven Business Process Modeling". In: Business Process Management, LNCS, vol 5240, 2008, pp. 84-99.

[12] Noy, N. and Musen, M.: "Algorithm and tool for automated ontology merging and alignment". In: Proceedings of the 17th National Conference on Artificial Intelligence, Aug. 2000, Austin, USA.

[13] Jablonski, S. and Goetz, M.: "Perspective oriented business process visualization". In: Business Process Management Workshops, LNCS, vol. 4928, 2008, pp. 144-155.

[14] Jablonski, S. and Bussler, C.: "Workflow management: modeling concepts, architecture and implementation". International Thomson Computer Press, London, 1996, ISBN 1850322228.

[15] Jablonski, S., Igler, M., and Günther, C.: "Supporting collaborative work through flexible process execution". In: 5th International Conference on Collaborative Computing, Nov. 2009, Washington DC, USA.

[16] Halleux, J.d.: "QuickGraph, Graph data structures and algorithms for .net". Last access: Mar. 2012. Available: http://quickgraph.codeplex.com..

[17] Microsoft: "Graph# - Layout algorithms and graph layout control for WPF applications". Last access: Mar. 2012. Available: http://graphsharp.codeplex.com.

[18] Verbeek, H., Buijs, J., Van Dongen, B., and Van der Aalst, W.: "XES, XESame, and ProM 6". In: Information Systems Evolution, Lecture Notes in Business Information Processing, vol. 72, 2011, pp. 60-75.

 ISBN: 978-1-61208-201-1