Process Observation as Support for Evolutionary Process Engineering

Stefan Schönig[1], Michael Seitz[2], Claudia Piesche[1], Michael Zeising[1], and Stefan Jablonski[1]
[1] University of Bayreuth
Chair of Applied Computer Science IV
Bayreuth, Germany
{stefan.schoenig, claudia.piesche, michael.zeising, stefan.jablonski}@uni-bayreuth.de

[2] PRODATO Integration Technology GmbH
Erlangen, Germany
michael.seitz@prodato.de

***Abstract* – The Process Observation project is used to generate business process execution logs and guides process participants through process execution. In this contribution, we introduce process evolution as an economic field of application for process observation. There are different needs for process evolution, e.g., to establish more consistent process results, continuously measure and improve process performance or meet accreditation requirements. We will show how process discovery, process guidance and process evidence as the main basic functions of process observation can be applied as support for reasonable process evolution. In this way, process observation can be used to reach a desired evolution stage or rather facilitate the transition between two maturity levels. Furthermore, process observation serves as an implementation for certain evolution stages itself and can additionally be consulted to prove the conformance to quality requirements of maturity levels.**

Keywords* – *Business Process Observation; Process Monitoring; Process Mining; Process Model Enactment; Process Evolution

I. INTRODUCTION

This article is an extension of a previously published paper [1]. 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 [2] [3]. 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 [4]. Since process modeling is an expensive and cumbersome task, we identify approaches that promise to reduce the modeling effort.

Furthermore, traditional process management restricts the support of process execution by strictly separating modeling and execution phase. This separation exemplarily flows into the following propositions [35]:

- Schematization of processes: everything, that has not been modeled, is excluded from management and will not be supported by systems.
- Inadequate consideration of exceptions: a priori models are often idealizations. Dynamically occurring exceptions will consequently not be supported by systems.
- By strictly separating modeling and execution phase process models are often incomplete because theory and practice are deviating.
- Inadequate feedback: generally, there is no consideration of feedback from process execution to process modeling. However, knowledge of previous executed process cases should be considered during process modeling.

Latest research achievements in the context of pattern discovery and data mining offer possibilities that promise to overcome the strict separation of modeling phase and execution phase by applying real-time analysis methods of executed processes.

One of these research achievements 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 [5]. 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 [6]. 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, a Process Observation (PO) tool, 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 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 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, PO must offer a stimulus that motivates process participants to work with PO. This stimulus is put into effect by a recommendation service. PO continuously analyzes available process log data to guide the process users. This means, it suggests process steps, documents or tools that the user most probably performed or used. 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 tool recommends possible process entities. This dynamic recommendation service becomes more and more reliable the more process instances have been executed under the guidance of 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.

In the following, we want to classify 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. 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 addition to our previous work of [1], we want to introduce process evolution as an economic field of application for PO in this article. There are different needs for process evolution, e.g., to establish more consistent process results, continuously measure and improve process performance or meet accreditation requirements. We will show how automatic process model generation (process discovery), dynamic recommendations (process guidance) and access to execution logs (process evidence) as the main basic functions of PO can be applied as support for reasonable process evolution. In this way, PO can be used to reach a desired evolution stage or rather facilitate the transition between two maturity levels. Furthermore, PO serves as an implementation for certain evolution stages itself and can additionally be consulted to prove the conformance to quality requirements of maturity levels.

This article shows the following structure: Section II introduces process evolution as an appropriate field of application. Section III introduces business process enactment and Section IV gives an overview over current existing process model enactment approaches. In Section V, we will explain the conceptual details and the general approach of process observation. Furthermore, concrete implementation techniques will be shown in Section VI. Based upon the introduced conceptual details of the previous sections, we will describe different applications for process evolution support and some use cases in Section VII. Section VIII describes the influence of the PO on the current process lifecycle. In Section IX, we will give an overview over related works. In Section X we will finally conclude and give an outlook on further research issues and applications.

II. Evolutionary Process Engineering

Before actually describing our approach in detail we want to introduce process evolution as an economic field of application for process observation.

A. *Introducing Process Evolution*

We perceive process evolution as a certain and natural development each process encounters over time in order to increase its quality and performance. There are different needs for process evolution, e.g., to expand business, establish more consistent and more stable process results, continuously measure and improve process performance or meet accreditation requirements. As an example, the workflow continuum used in [38] can be cited. It shows how a process develops from a unique, exceptional use case (e.g. new type of customer request) to the point of a highly recurrent, well-structured standard workflow. From a quality management perspective, adequate measurements for process evolution are maturity models, such as the Capability Maturity Model Integration (CMMI) [43] or the Business Process Maturity Model (BPMM) [42]. As stated in [37], the evolution of quality is accompanied by (new) requirements to the process model and its enactment for process support. While a low maturity level is satisfied by rudimental modeling of the expected results aiming at a human controlled process execution, a higher maturity level probably requires a formal specification of the organizational and operational structure in order to implement a (partly) system controlled process execution. Anyway, [37] insists on fitting both process model and process support to the quality and performance requirements in order to drive the process in a reasonable way. For example, when it does not make any difference for the success of the process in which way a required document is created it is not worth the effort to design a process model prescribing the user the order of each single step, because this kind of execution support would not be accepted by the users.

In the following section, the maturity levels including an ideal set of climactic evolution stages are described and the requirements for an adequate support of process evolution through the PO are derived.

B. Requirements for Process Evolution Support

The maturity model for process evolution in [37] contains five maturity levels (ML1 - ML5) and is based on the perspective oriented process model (POPM). According to POPM, processes can be described by at least five independent perspectives [14]: the functional perspective (work steps), the behavioral perspective (control flow), the data oriented perspective (data), the organizational perspective (people) and the operational perspective (tools). In order to get a general idea about POPM perspectives, we recommend [14] and [15]. Each maturity level in [37] describes a suitable and stable stage of process evolution through dedicated characteristic values of the dimensions process quality, process model and process support. Below, the maturity levels are summarized. Afterwards, requirements for PO to support process evolution are derived.

For ML1 it is sufficient to design a result-oriented process model that covers a list of necessary steps or expected results respectively (functional perspective). This information should be published as support for process participants. It has to be documented properly that the process has actually been executed and the results could be achieved.

According to ML2 the process model should represent a project plan and therefore additionally contain resources and responsibilities (organizational perspective) and a time schedule (behavioral perspective). In order to support the execution at runtime plan deviations should be recognized and communicated. It has to be proved with the help of documentation that the process has really been projected and monitored.

ML3 implies a standardization of the process and accordingly is in need of a reference process model that moreover must include information about the usage and invocation of mandatory tools, applications and services (operational perspective) as well as the default document format and structure (data perspective). During execution the active monitoring of the compliance has to be enabled and supported. The compliance with the standard process is required and must be proved by documentation.

As for ML4, key performance indicators have to be defined, measured and analyzed. Therefore, a formal and (technically) comparable process model with metering points is required that enables the extraction of consistent and stable process logs. The execution has to be supported by an adequate analysis and control tool. It has to be proved that the process targets are actually monitored statistically across several process instances.

ML5 demands the establishment of a systematic continuous improvement process. It necessitates a continuously formal and extensible representation of all occurring process perspective contents across the whole process lifecycle. A system is needed for execution support that allows for respective improvement measures at runtime and is able to reuse them. It has to be proved through documentation that improvements are (probably automatically) incorporated into the reference process model and also implemented successfully in future instances.

It is clearly obvious that with each maturity level a more comprehensive and more detailed process model is needed. Thus, process model engineering can be regarded as an important factor of process evolution and thus constitutes one requirement for its support. After all, the designed process model is used to provide appropriate runtime support. The enactment of process models serves as implementation of evolution stages and is intended to enable the establishment of the desired quality as efficient as possible. Therefore, enactment is considered as further requirement for process evolution support. A maturity level can first be regarded as fulfilled if the demanded quality can be proved. Consequently, documentation of the way the process has actually been executed is yet another requirement for process evolution support.

III. ENACTMENT APPROACHES OF PROCESS MODELS

In the previous section, engineering, enactment, and documentation could be identified as requirements for adequate process evolution support.

Besides engineering and documentation purposes, the enactment of process models is one main target of business process modeling efforts. So it is no wonder that various research activities center on questions on how to best support enactment of process models [14], [15], [24]. Even though most publications deal with particular implementations for process execution support or workflow management systems (WfMS) at it one can recognize the broad relevance of process enactment as concept within the BPM research context. Aside from different presuppositions, i.e. imperative vs. declarative approach, limitation to the execution order vs. perspective-oriented modeling or consideration of business process as serialization of web services, the overall goal is the best possible execution support for processes within organizations. However, the main questions remaining are: How does this support look like? Which type of support is considered as best fitting for a particular situation? Which conceptual approach provides best results regarding efficiency of process execution, quality of resulting products or employee satisfaction?

A first rough answer to these questions can be found for example in [23], where additional to process modeling also process execution issues are discussed and possible solutions are introduced. Similar to database management, the execution of processes is compared with the usage of stored data. Just as the data usage differs immensely between various application scenarios, i.e., "even two different implementation concepts are necessary (normal databases vs. warehouses)", the same observation is valid with process management. Therefore, the conclusion can be drawn, that different use cases for processes need different enactment approaches. To show this is the main contribution of this paper.

A. *Definition of Business Process Enactment*

From a historical perspective, process management and especially process enactment originates from office automation [25] as well as from the concept of BPEL (Business Process Execution Language) where a business process is defined as "a large-grained Web Service which executes a control flow to complete a business goal" [24]. In the same way, the classical workflow management system follows the concept of strictly automated workflow according to an explicit process model. Even [25] regards the enactment of processes only as the generation of corresponding runtime support based on a process model. However, there are example processes which cannot follow this assumption, for example creative processes. Here, the process model rather presents a guideline how to proceed; the sequence of process execution can more or less freely be chosen whereby the process model functions as a "recommended guideline".

Thus, enactment must be seen in a broader sense. It shall not only support a strict default workflow but offer recommendations instead. It happens mainly through use of a process, i.e. enactment is not the same as strict execution, rather more similar to "process usage". It aims at applying or using processes and instantiates a process model in any way, automatically by means of various IT systems (explicit) or through the specific work process of employees within a company (implicit). Therefore, various kinds of support while conducting a process instance are conceivable. Such approaches vary between mere manually controlled and fully automated enactments. Thereby, simple wallpaper models (which is nothing but the printed out process model) with check marks define one end of the spectrum; automatic process execution with WfMS demarcate the other end.

B. *Motivation and Contribution*

As stated before, there is a growing amount of workflow and process enactment approaches, especially considering the broader definition of enactment. However, until now there is a lack of a conceptual evaluation of the quality and adequacy of these approaches. At the same time, enactment is considered an important phase of the business process life cycle [25], [26]. Even though the modeling phase can be seen as the more important part of process management, the implementation of processes gains in importance and deserves closer attention.

The contribution of this section at first consists in widening the view on enactment approaches by introducing a broader definition getting beyond the current limited view of enactment as an automatically supported activity. It is shown that enactment support can be achieved by IT systems as well as by other manual approaches.

IV. EXISTING ENACTMENT APPROACHES

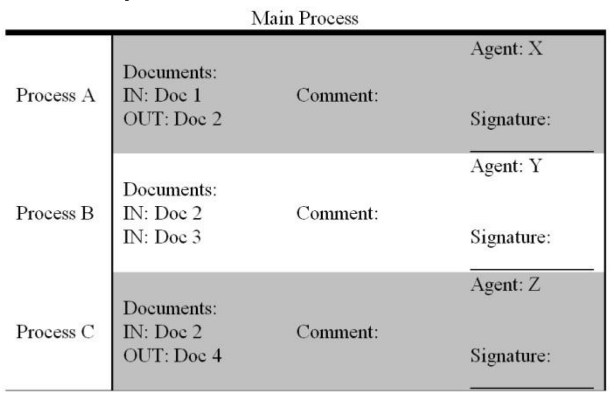
This section enfolds the spectrum of enactment approaches from manually controlled process execution to fully IT-supported ones. Thereby, the order of described approaches is somewhat equivalent to the increasing automation of process execution, i.e., external tracking and serialization can manually be carried out but the other three approaches need sufficient support by IT systems. However, the enumeration of following approaches depicts an exemplary selection of enactment approaches and is not exhaustive. The objective of this section is to put in context the concept of Process Observation as Dynamic Guidance approach.

A. *Model-as-is*

Inspired by documentation purpose of process models, external tracking is aiming at supervising the process on an abstract level. Thereby, the model is used as is, i.e., the model is printed out and used as wallpaper or it appears as a map electronically. Responsible agents may mark finished tasks by labels and with it retain control over process execution.

B. *Checklist*

The serialization of process steps happens by defining a checklist that "comprises the main process steps including documents that must be produced and agents that are responsible to perform the corresponding process" [23]. Every process has to be signed after finishing the execution so that, finally, the process is concluded if all steps are signed. A checklist serializes the process steps of a process model, shows what input sources could be used, what results are expected, and who is responsible for the step. The method can be accomplished manually as well as electronically.



Main Process

Process A	Documents: IN: Doc 1 OUT: Doc 2	Comment:	Agent: X Signature:
Process B	Documents: IN: Doc 2 IN: Doc 3	Comment:	Agent: Y Signature:
Process C	Documents: IN: Doc 2 OUT: Doc 4	Comment:	Agent: Z Signature:

Figure 1. Sample Checklist

C. Dynamic Guidance

Aim of this enactment approach is to guide users through a flexible process instance at runtime based on the process model but not restricted to it, i.e. without patronizing the user. Thus, the responsible agent has the possibility to change the order of tasks due to his experience, extraordinary conditions or optimization purposes.

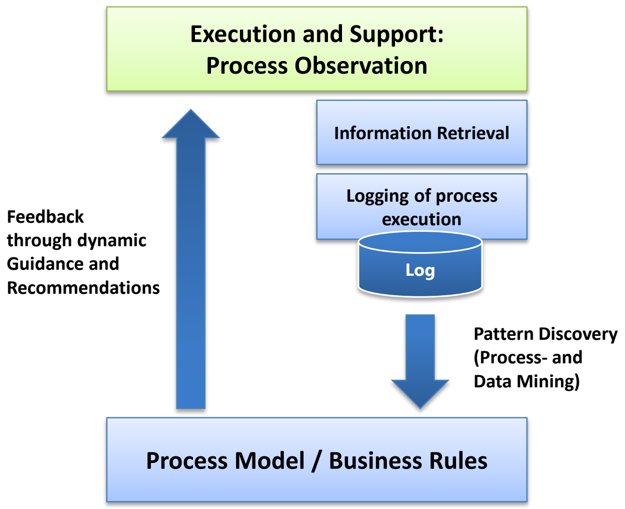

Execution and Support: Process Observation
Information Retrieval
Logging of process execution
Log
Feedback through dynamic Guidance and Recommendations
Pattern Discovery (Process- and Data Mining)
Process Model / Business Rules

Figure 2. Concept of the Process Observer

The idea for dynamic guidance relies on data- and process mining approaches. Event logs of former process executions are used to find recommendations that are displayed while current process execution within a process instance. The conceptual basis of dynamic guidance is stated in [27] and [40]. It describes an example for the implementation of dynamic guidance using process mining techniques in order to be able to examine various perspectives of process execution logs. The algorithms take the current process scenario into account and offer a best practice recommendation to the user. In contrast to static algorithms, process mining does not need a predefined aim in order to calculate a recommendation. Nevertheless, this approach requires access to log files. However, there are processes that are not electronically executed and controlled. In such cases, the generation of manual logs has to be initiated by process participants by recording their current activities. This information will be enriched and complemented by optional logs from information systems. An example for a dynamic guidance implementation is the Process Observation Project stated in [1]. The conceptual details of this project are presented in Section V and further implementation details are presented in Section VI of the work at hand.

D. Workflow Management System

The Workflow Management Coalition (WfMC) defines a Workflow Management System as: “A system that completely defines, manages and executes “workflows” through the execution of software whose order of execution is driven by a computer representation of the workflow logic.” [28] Insofar, WfMSs are reduced to the strict execution of workflow logic represented by more or less complete process models. They provide support in three functional areas:

- Build-time functions concerned with defining and modeling the workflow process
- Run-time control functions concerned with managing the workflow processes in an operational environment and sequencing the various activities to be handled as part of each process
- Run-time interactions with human users and IT application tools for processing the various activity steps

Likewise, [14] refers to workflow management systems as consisting of two parts (build time and run time). “The build time part allows a modeler to define and maintain all the information necessary to eventually execute workflows. The workflows are finally performed by the run time part of the workflow management system.” Thus, only the run-time functionalities (Workflow Enactment Service) lay within the scope of this paper, are therefore examined in detail and are stated by the workflow reference model as follows [28]:

- interpretation of the process definition
- control of process instances - creation, activation, suspension, termination
- navigation between process activities that may involve sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data
- sign-on and sign-off of specific participants
- identification of work items for user attention and an interface to support user interactions
- maintenance of workflow control data and workflow relevant data, passing work-flow relevant data to/from applications or users
- interaction with external resources (client or invoked application)

One well-known WfMSs today is Declare [29] based on a constraint-based approach and offers with it support for loosely-structured processes.

Within the last decade of research activities on process management a "transition from workflow management systems focusing on automation of structured processes to Business Process Management (BPM) systems aiming at a more comprehensive support of a wider variety of business processes" [25] can be observed. "BPM extends the traditional WFM approach by support for the diagnosis phase […] and allowing for new ways to support operational processes […]" [26]. Because this enhancement does not affect the results of this paper, the term WfMS is used here, but with keeping in mind the concept of business process management systems.

E. Process Navigation System

During the last years, the increasing demand for so-called flexible WfMSs has caused a rethinking on IT-supported process enactment. A flexible business process is understood to restrict participants to a lesser extent [34] and must therefore comprise more possible paths than a rigid one.

In this context, two paradigms are distinguished: the imperative and declarative style of modeling business processes. For an imperative model, every possible path must be foreseen at design (build) time and encoded explicitly. A missing path is considered not allowed. Typical imperative languages include the Business Process Model and Notation (BPMN), Event-Driven Process Chains (EPC) and UML Activity Diagrams. In declarative modeling, on the contrary, only undesired paths and constellations are excluded so that all remaining paths are potentially allowed and do not have to be foreseen individually. The declarative way of modeling is considered best suited for the flexible type of business processes [30].
Declarative modeling is based on constraints that relate events of the process case and exclude or discourage from certain correlations. We argue that both constraints and events must be able to involve all relevant perspectives of business process like, e.g., incorporated data, agents performing the work and utilized tools [31]. On this way it becomes possible to express realistic correlations like, e.g., the actual performing agent of a process step affecting the type of data used in another step [31].
Besides turning to declarative cross-perspective modeling, a Process Navigation System must support different modalities for differentiating between optional and mandatory constraints. Thus, process modelers may distinguish easily between the hard boundaries of a process (mandatory constraints) and best practices (optional constraints) and are therefore able to formulate business process models of a very high expressivity.
Flexible processes usually involve human participants. A Process Navigation System enacting these human-centric processes should therefore be treated as a decision support system instead of a means of automation. As a consequence, an according system should propose actions and support them but it should never enforce them [32]. This requirement goes along with the call for process management systems "to provide directions and guidance rather than enforcing a particular route" whereas they are compared to navigation systems [39]. An important characteristic of decision support is explanation and so-called meta-knowledge [33]. Proposals made by the system need to be justified and explained so that sound choices can be made. For process execution, this means that certain proposed actions must be marked as recommended and that discouraged actions can be traced back to and explained by the according parts of the business process model. This characteristic of a Process Navigation System will be referred to as traceability.

An execution engine that aims at fulfilling the above requirements and therefore allows for declarative, cross-perspective and multi-modal process models and a traceable execution is being developed [41].

V. Process Observation and Guidance through Process Execution

This section is mostly part of our previous work of [1]. 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. 3):

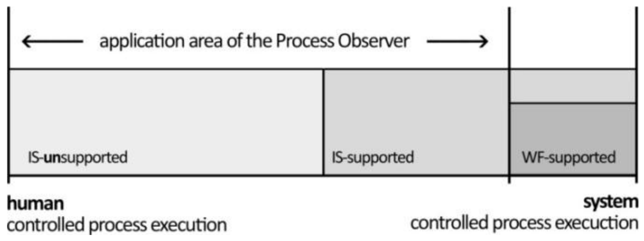

application area of the Process Observer
IS-unsupported
IS-supported
WF-supported
human
controlled process execution
system
controlled process execution

Figure 3. 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) [16]. 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, PO turns its attention to IS-supported and IS-unsupported processes (Fig. 3). 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 Observation Project*

The challenge of 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, PO must offer a stimulus that motivates process participants to support manual logging. Since 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). 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 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. 3, 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 PO offers such guidance. It consists of two sub-features: dynamic recommendations and auto-suggest function.

1) *Dynamic Recommendations:*

Dynamic recommendations are generated in the following way. After the completion of a process step, 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, 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 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. 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. 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 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 subsection 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 PO. In order to handle problems like this, 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. 4). After defining a mapping between processes, a repeated execution of process mining results in the visualization of the amended process model.

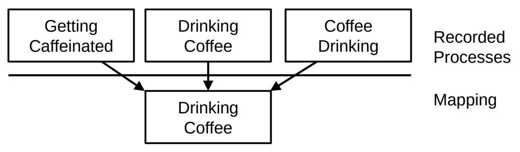


Figure 4. Sample mapping of recorded processes

VI. Architecture and Implementation

The following section mostly relies on our previous work of [1]. In this section, we will describe the architecture and implementation of 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 Observation Logging Client*

The core of 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., PO can be used in almost all applications. If the users are working in a "normal" office, PO can run on a stationary PC or notebook, if users are working "in the field", 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 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. 5). 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.

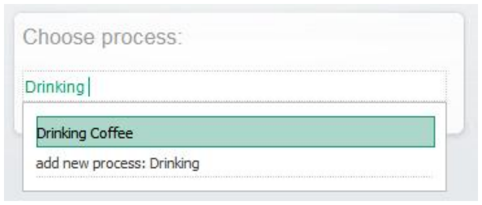


Figure 5. Example of auto suggestion list

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

As already described in Section V, 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 [4] 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 [17]. For implementation details concerning the alpha algorithm we refer to [4]. 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, 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 (Fig. 6).

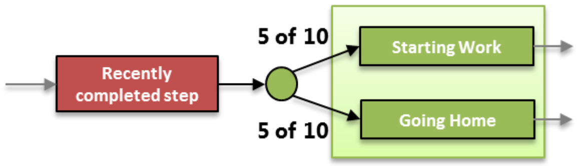
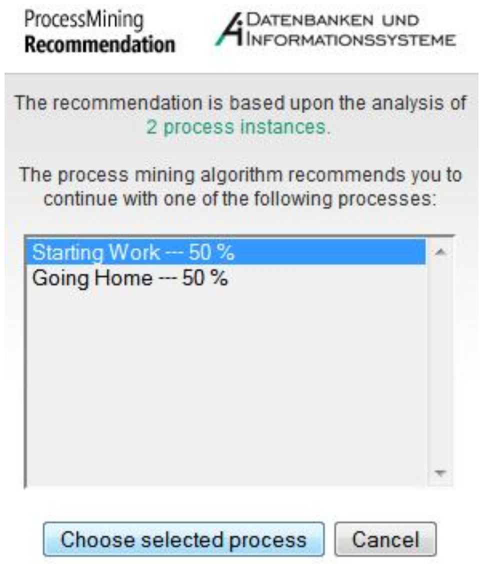


Figure 6. Example of an AJAX modal popup dynamic recommendation

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 (Fig. 7), giving the user the possibility to choose the following process step.


ProcessMining Recommendation
Datenbanken und Informationssysteme

The recommendation is based upon the analysis of 2 process instances.

The process mining algorithm recommends you to continue with one of the following processes:

Starting Work --- 50 %
Going Home --- 50 %

Choose selected process
Cancel

Figure 7. Example of an AJAX modal popup dynamic recommendation

C. *Administration interface*

Additionally, 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 V. 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 [18] 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 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.

VII. Using Process Observation to Support Evolutionary Process Engineering

In this section, we will show how process discovery (i.e. the automatic generation of process models based upon event logs), process guidance and process evidence as the main basic functions of PO can be applied as support for reasonable process evolution. The presented examples und use cases refer to the maturity levels for process evolution described in Section II.

A. *Process Observation Applications for Process Evolution Support*

PO can be applied in three respects to support process evolution:

1) *Engineering*

Firstly, PO can be used to reach a desired evolution stage or rather facilitate the transition between two maturity levels by enhancing the process model (process discovery) conveniently. The engineering function accomplished by the PO is to gradually attain missing parts or details of particular process perspectives according to POPM that are needed to comply with the target maturity level.

Example: ML2 requires a time schedule. In order to engineer the behavior perspective PO inquires the executed process steps over a specific period. In combination with timestamp information the PO derives execution order and dependencies. The attained process model can now be enacted by a project management system and e.g. visualized through a Gantt chart.

2) *Enactment*

Secondly, PO serves as an implementation for certain evolution stages. Among other information systems, PO provides execution support (dynamic guidance, recommendations) through enactment of extracted process models (see Section V) and thus can be used systematically to meet the requirements for process support as prescribed by the maturity levels.

Example: PO can provide flexible execution support for ML3 by suggesting valid process steps and applicable tools according to the reference process. Since the user himself decides what he actually does, PO cannot prevent deviations from standard but advise him of not being compliant after comparing the event logs with the reference model. Additionally, the user is guided through the process by suggesting best-practice rules as visualized in Fig. 7.

Which maturity levels are further covered by PO and what are other possible enactment approaches is clarified by the two use-cases in the following subsections B and C.

3) *Documentation*

Thirdly, PO is able to gather information about the way processes have actually been executed and therefore can be consulted to prove the conformance to the quality requirements of the maturity levels (process evidence). Since PO has access to both manual and automatic logs, it is able to provide complete process execution event logs as documentation and audit trail even if the process is partly executed beyond the control of any information system.

Example: IT management is about to analyze the information system landscape and therefore desires a consolidated documentation and quantity structure across all processes about the usage of tools and services (operational perspective) with relation to the respective professional need (functional perspective). Since existing automatic logs are not sufficient, the PO is recording the usage of tools and services as for the “offline” processes that are not provided with any process log or documentation. Through merging these manual logs with the automatic logs a cross-process reporting view is established.

B. *Use-Case 1: From ML1 to ML2*

Initial situation: The process results are achieved somehow without the help of any IS. The process steps are visualized on wallpaper. It serves as orientation for the participants (ML1).

Target situation: Proper results should be achieved in time (ML2). A project management system (PMS) is intended to support both project manager (PM) and participants. The PM should be provided with up-to-date information about the progress in order to be able to monitor and control the process.

Engineering: The first use case comprises the generation of a manual log (Fig. 8) without an information system being available. The participating agents are executing the corresponding processes under the guidance of PO. The manual log is finally analyzed by process mining algorithms. The resulting process models can be fed into a PMS if wanted and if available. Thus, process models can afterwards be enacted by a PMS.

Both order of the process steps and time flow are engineered by PO through mining of manual logs. Beside the logged in user or role, PO is able to gather start, end and abortion timestamps. When a user declares he is about to start a new step, it could also be possible that PO additionally asks him when it will presumably be completed. By this means, all required runtime information for comparison with the schedule is made available. Already during the engineering process the project manager is able to gather the current project status by consulting the PO logs and align this information with wallpaper.

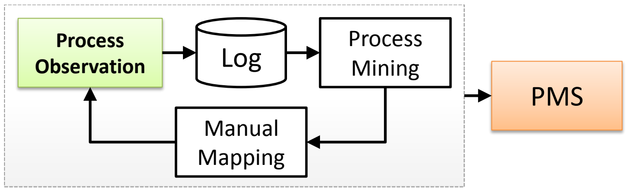


Figure 8. First use case – generation of manual logs

Enactment: We assume that after the transition from ML1 to ML2 the wallpaper is not needed any more and execution support including recognition and communication of project plan deviations (as demanded by ML2) is finally accomplished by the project management system. Therefore, the process structure attained through process discovery serves as project plan template and is enacted by the PMS. Actual due dates can be added manually by PM. At runtime, the progress information is either permanently updated by the PO automatically or maintained by the users and the project manager manually. In both cases the project plan update has to be attained through user interaction.

Documentation: Through combination of manual and automatic logs provided by the PO and PMS it will succeed to prove that the process is managed in such a way that proper results can be achieved in time and thus complies with ML2.

C. Use-Case 2: From ML4 to ML5

The second use case comprises the application of PO in parallel to an information system (Fig. 9).

Initial situation:

Process execution is already supported by a fully-fledged WfMS. However, traditional WfMS do not allow for deviating from enacted process models. That's why process participants have to build workarounds in order to be able to execute process steps that had not been considered during process modeling phase, e.g., exceptions or newly occurring process cases. From time to time, process managers have to discuss, probably on the basis of statistically measured KPIs, if the process can be improved somehow (ML4). Thereby, it will be reviewed, if enacted process models had been an adequate basis for the running processes or if additional use-cases or exceptions that had not been considered in process models yet should be introduced in order to improve the performance. Therefore, process managers have to adapt process models manually according to the results of the discussions. Note, that this situation has already been described in the introduction of this article. Traditional process management restricts the support of process execution by strictly separating modeling and execution phase.

Target situation:

If standard process models are not suitable or if a different way promises better results, process participants should have the possibility to deviate from standard during process execution without losing control and overview. There should be an adequate consideration of exceptions: dynamically occurring exceptions should be supported by systems. Furthermore, there should be a systematical support of continuous process improvement (as demanded by ML5).

Engineering:

The research achievements of the previous sections, i.e., Process Observation, offer possibilities to overcome the strict separation of modeling phase and execution phase by applying real-time analysis methods of executed processes. Dynamically occurring exceptions (that are not yet part of enacted process models) are engineered by PO through manual logging and subsequent process mining. On this way, it is possible to overcome the strict separation of modeling and execution phase and process models can be completed automatically. Newly occurring process steps or exceptions are discovered by PO and added to the resulting enhanced process models. Furthermore, there is an adequate feedback: with PO, it is possible to consider feedback from process execution to process modeling. The knowledge of previous executed process cases could be added to process models as recommendations, e.g., which path through the model was the fastest or cheapest one.

Enactment:

We assume that after the transition from ML4 to ML5 the process model enactment of a running WfMS is featured by the application of PO. The intention is to complete the process execution log information mutually. Identical processes are merged to one single process. Process Mining is finally applied to the joint log (Fig. 9). Identified processes can be fed back into information systems. On this way, the resulting process models are enhanced by newly occurring process steps and recommendations. The joint application of WfMS and PO provides a proper solution for process improvement as demanded by ML5.

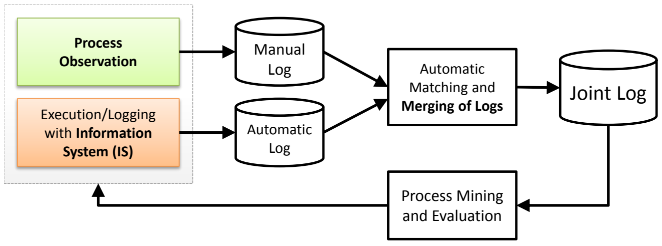

Process Observation
Execution/Logging with Information System (IS)
Manual Log
Automatic Log
Automatic Matching and Merging of Logs
Joint Log
Process Mining and Evaluation

Figure 9. Second use case – merging of logs

Documentation:

The joint application of a WfMS and PO provides a process execution log that contains execution information about process steps that have been executed with support of WfMS as well as under the guidance of PO. On this way, the joint log provides evidence for continuous process improvement. Therefore, we define a value *compliance_level* (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 a matching algorithm identified two processes as equal, the numerator *#compliance_level* will be increased by 1. After finishing traversing both log files, the resulting value of *#compliance_level* is divided by the total number of recorded processes within the manual log.

$$compliance\_level = \frac{\#matched\_processes}{\#recorded\ process\ with\ PO} \qquad (1)$$

Like this, the calculated value reflects how many processes are already executed with support of the WFMS. For a special organization a *compliance_level* value of 0.9 may be enough. This means, 90% of the executed processes are implemented and supported by the WFMS. Like this, the value reflects the continuous improvement of enacted process models.

VIII. CHANGES WITHIN THE PROCESS LIFECYCLE THROUGH THE APPLICATION OF PROCESS OBSERVATION

In this section, we will describe the impact of the application of PO on different phases in the process lifecycle. As already mentioned, the previous process lifecycle [2] 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 [4], 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. 10).

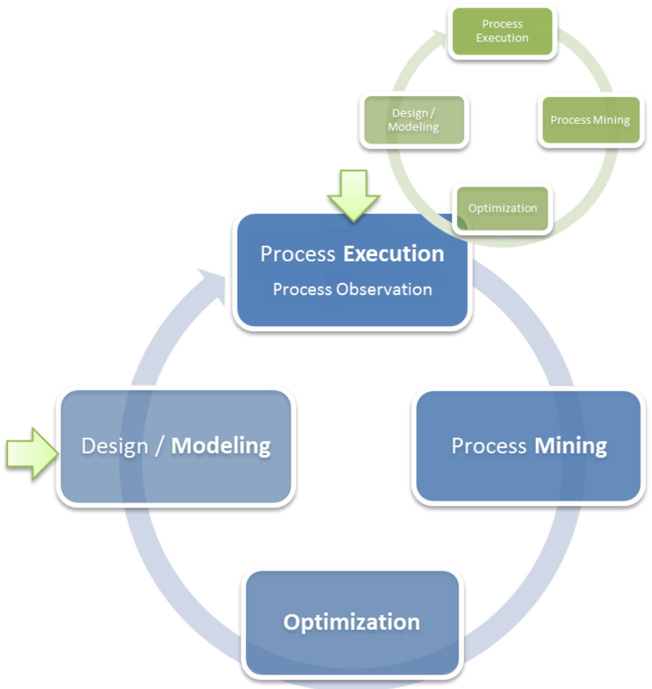


Figure 10. Adapted process lifecycle through the application of PO

The new lifecycle contains two different entry points. In addition to the traditional initial modeling phase, the process lifecycle can be entered by process execution (as usual) accompanied by manual logging, i.e., the generation of a manual log, with PO. Note, that the execution phase consists of a lifecycle in little again: after the execution of every process step, the process observation tool analyses the existing process execution log by process mining methods. The mining results are directly sent back to the process participant who has the possibility to react on the results dynamically during the running process (online process observation). On this way, it is possible to learn from previous mistakes on runtime (Optimization) and to change ones behavior, i.e., the path through the process (Modeling). Having executed a desired number of process instances (cases), the process observation tool can be used to extract process models that can be enacted in other process enactment approaches as stated in Section IV. Here, the results of process mining possibly have to be reworked in a process remodeling phase (Optimization). The benefit of the application of PO consists of the time saving between the previous process modeling phase and the less time consuming remodeling phase.

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 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 steps, business processes can be evaluated and finally optimized.

IX. Related Work

The last decades of research activities concerning business processes led to a growing amount of process enactment approaches, including a great variety of types of enactment support as well as many different supporting implementations.

Many publications are engaged in the comparison of particular implementations of WfMS and BPMS respectively. [20] offers a good market overview by evaluating the top 25 vendors in the BPMS market. The results are displayed in the so-called Magic Quadrant, spanned by dimensions ‘ability to execute’ and ‘completeness of vision’. Each vendor will therefore correspondingly be classified as niche player, visionary, challenger or leader. However, this evaluation considers only systems fulfilling economically oriented criteria. This approach is a practical one and does not satisfy research interests. Some further efforts like the pattern-based evaluation of scientific WfMS [21] or the evaluation of workflow management systems using meta models [22] examine implementations in a more scientific way and without the use of scoring strategies. For example, [22] introduces a meta model approach for the evaluation of WfMS. A methodology for the selection of workflow products is specified as done by well-known standardized scoring models for software evaluation. This kind of evaluation can only be applied to build-time components used for modeling of organizational entities and the creation of the process model. Furthermore, there must be a high degree of formalization on the user side, because business situations have to be depicted using formal methods.

Even though there are some promising approaches concerning the evaluation of enactment-related implementations, the approaches still remain comparisons of particular implementations. The article at hand stands in the tradition of research in [23] trying to answer questions on: What does “enactment” mean? What alternative enactment approaches can be distinguished?

Besides workflow management system, this article defines various existing types of enactment support, i.e., so-called *model-as-is*, *checklists*, *dynamic guidance* and *process navigation systems*. In this regard, the paper aims at getting over the current limited view of enactment as an automatically supported activity.

The work at hands introduces process evolution as an economic field of application for Process Observation (PO) [1] in order to achieve a higher degree of maturity. In order to support evolutionary process engineering, one of the basic functions of PO is enabling automatic process discovery.

The idea of automating process discovery through event-data analysis was first introduced by Cook and Wolf in the context of software engineering processes [7]. 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 [6]. Generally, the goal of process mining is to extract information about processes from event logs of information systems [8]. There are already several algorithms and even complete tools, like the ProM Framework [9], 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 [6] and [8]. For the first prototype of PO, we used the alpha-algorithm of [4] and the heuristics-miner [10].

However, for our future research activity we consider declarative process mining algorithms like [36] appropriate. Declarative process modeling techniques offer the possibility to describe complex applications by comprehendible process models. In contrast to imperative modeling, declarative models concentrate on describing what has to be done and the exact step-by-step execution order is not directly prescribed.

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 by capturing data on personal task management [11] [12]. However, these approaches are not process based. They are not analyzing execution orders and therefore it is not possible to extract adequate process models that can finally be enacted in process management systems.

In order to discover identical processes between different data storages, we suggest using basic automatic ontology matching algorithms [13]. 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 [2]. 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 [4]. 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 POPM.

X. Conclusion and Outlook

In this article, we introduced process evolution as an economic field of application for process observation. We showed how process discovery, process guidance and process evidence as the main basic functions of process observation can be applied as support for reasonable process evolution. Furthermore, we introduced the spectrum of process model enactment approaches and put in context the concept of Process Observation as Dynamic Guidance approach. Process observation serves as an implementation for certain evolution stages itself and can additionally be consulted to prove the conformance to quality requirements of maturity levels.

Our future research activity in the field of Process Observation will start with the integration of declarative process mining methods and other data mining approaches like association rule mining in order to increase the quality and understandability of extracted process models and the power of the recommendation module. Furthermore, we will include information retrieval methods and implementations like search engines in order to facilitate process information provision. Additionally, we need to include matching methods in order to match and merge identical processes. 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. Finally, we are looking forward to an extensive application of the PO in an organization, accompanied by a detailed documentation of the practice.

Acknowledgment

The presented system is developed and used in the project “Kompetenzzentrum für praktisches Prozess- und Qualitäts-management”, which is founded by “Europäischer Fonds für regionale Entwicklung (EFRE)”.

References

[1] Schönig, S., Günther, C., and Jablonski, S., "Process Discovery and Guidance Applications of Manually Generated Logs," presented at the The Seventh International Conference on Internet Monitoring and Protection (ICIMP 2012), Stuttgart, Germany, 2012.

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

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

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

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

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

[19] 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.

[20] Sinur, J. and Hill, J. B., "Magic Quadrant for Business Process Management Suites," Gartner Research ID Number: G00205212, 18.10.2010 2010.

[21] Migliorini, S., Gambini, M., La Rosa, M., and ter Hofstede, A. H. M., "Pattern-Based Evaluation of Scientific Workflow Management Systems," ed, 2011.

[22] zur Muhlen, M., "Evaluation of workflow management systems using meta models," in System Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on, 1999, p. 11 pp.

[23] Jablonski, S., "Do We Really Know How to Support Processes? Considerations and Reconstruction," in Graph

Transformations and Model-Driven Engineering. vol. 5765, G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, Eds., ed Berlin, Heidelberg: Springer 2010, pp. 393-410.

[24] Pesic, M., "Constraint-based workflow management systems:shifting control to users," Technische Universiteit Eindhoven, Eindhoven, 2008.

[25] ter Hofstede, A. H. M., v. d. Aalst, W. M. P., Adams, M., and Russell, N., Modern Business Process Automation: YAWL and its Support Environment. Heidelberg: Springer, 2010.

[26] van der Aalst, W. M. P.: "Business process management: a personal view," 2004.

[27] Günther, C., Schönig, S., and Jablonski, S., "Dynamic Guidance Enhancement in Workflow Management Systems," presented at the The 27th Symposium On Applied Computing (SAC 2012), Riva del Garda, Italy, 2012.

[28] Hollingsworth, D., "The Workflow Reference Model," Workflow Management Coalition1995.

[29] Pesic, M., Schonenberg, H., and van der Aalst, W. M. P., "Declare: Full support for loosely-structured processes," 2007, pp. 287-287.

[30] Fahland, D., Lübke, D., Mendling, H., Reijers, H., Weber, B. and Weidlich, M., “Declarative versus Imperative Process Modeling Languages: The Issue of Understandability”. In: Enterprise, Business-Process and Information Systems Modeling (10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009), 2009, pp. 353-366.

[31] Igler, M., Faerber, M., Zeising, M. and Jablonski, S., “Modeling and Planning Collaboration in Process Management Systems using Organizational Constraints”. In: The 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010), Chicago, IL, USA, 2010, pp. 1-10.

[32] Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P. and Zbyslaw, A., “Freeflow: Mediating Between Representation and Action in Workflow Systems”. In: ACM Conference on Computer Supported Cooperative Work (CSCW '96), Boston, MA, US, 1996.

[33] Turban, E., Sharda, R. and Delen, D., “Decision Support and Business Intelligence Systems”, 9 ed., Prentice Hall, 2010.

[34] S. Sadiq, W. Sadiq, and M. Orlowska, "Pockets of Flexibility in Workflow Specification," in 20th International Conference on Conceptual Modeling (ER'2001), Yokohama, Japan, 2001.

[35] Vanderhaeghen, D., Fettke, P., "Organisations- und Technologieoptionen des Geschäftsprozessmanagements aus der Perspektive des Web 2.0". In: Wirtschaftsinformatik, no.1, 2010, pp. 17-32.

[36] Schönig, S., Günther, G., Zeising, M., Jablonski, S., Discovering Cross-Perspective Semantic Definitions from Process Execution Logs. In: Proceedings of the Second International Conference on Business Intelligence and Technology (BUSTECH 2012), Nice, France. ISBN: 978-1-61208-223-3.

[37] Seitz, M.; Jablonski, S., "Evolutionäres Prozess-Engineering: Zum angemessenen Grad an Prozessunterstützung" in HMD - Praxis der Wirtschaftsinformatik, vol. 287, D. Ingenhoff, A. Meier, ed: dpunkt Heidelberg, 2012.

[38] Huth, C.; Erdmann, I.; Nastansky, L., "GroupProcess: Using Process Knowledge from the Participative Design and Practical Operation of ad hoc Processes for the Design of Structured Workflows", 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9, 2001.

[39] van der Aalst, W.M.P., "TomTom for Business Process Management (TomTom4BPM)," in Advanced Information Systems Engineering. vol. 5565, P. van Eck, J. Gordijn, and R. Wieringa, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 2-5.

[40] Schönig, S., Zeising, M., Jablonski, S., “Adapting Association Rule Mining to Discover Patterns of Collaboration in Process Logs”. In: 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012), Pittsburgh, PA, USA, 2012.

[41] Zeising, M., Schönig, S., Jablonski, S., “Improving Collaborative Business Process Execution by Traceability and Expressiveness.” 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012), Pittsburgh, PA, USA, 2012.

[42] Object Management Group (OMG), “Business Process Maturity Model (BPMM)”, Version 1.0 (June 2008). Last Access: Sep 2012. Available: www.omg.org/spec/BPM/1.0/.

[43] Software Engineering Institute of the Carnegie Mellon University, “Capability Maturity Model Integration (CMMI)”. Last Access: Sep 2012. Available: www.sei.cmu.edu/cmmi/.