Comprehensive Business Process Management through Observation and Navigation

Stefan Schönig, Michael Zeising, and Stefan Jablonski

Applied Computer Science IV, University of Bayreuth, Germany
{stefan.schoenig, michael.zeising,
stefan.jablonski}@uni-bayreuth.de

Abstract. Most real-world business processes involve a combination of both well-defined and previously modelled as well as unforeseen and therefor unmodelled scenarios. The goal of comprehensive process management should be to cover all actually performed processes by accurate models so that they may be fully supported by IT systems. Unmodelled processes can be observed by the *Process Observation* system which generates models reflecting the recorded behaviour. Modelled processes may be of different natures: while so-called "automation" processes involve little human participation and mainly orchestrate services and applications, so-called "knowledge-intensive" processes are based on human expert participation. Both types of models may be enacted by the *Process Navigation* system. This contribution introduces the integration of both systems which leads to an approach for supporting the full range from unmodelled processes to both automation and knowledge-intensive processes as well as the transition from unmodelled to modelled processes.

Keywords: Business processes, workflow, process observation, declarative process modelling, process mining.

1 Introduction

Business process management (BPM) is considered an essential strategy to create and maintain competitive advantage by modelling, controlling and monitoring production and development as well as administrative processes [1, 2]. Many enterprises and organizations adopt a process model-based approach to manage their operations. Ideally, this involves surveying and modelling the as-is processes and designing to-be processes in a way so that they may be supported by BPM technologies. In reality, areas remain in which the as-is process may not be documented in a formal way so that they may not be supported by traditional workflow systems. As a consequence, these areas are then excluded from IT support. However, it is desirable that these unmodelled processes are supported by an IT system as well and that, ideally, this system offers suggestions for modelling these processes. This paper presents an approach for fluently covering both situations: the execution of modelled business processes and the support during unmodelled situations with a reconstruction of the actu-

ally performed process. Figure 1 gives an overview over the structure of the approach. The Process Observation (PO) system covers the support of unmodelled processes. Based on collected execution information, best practice patterns are provisioned to users. These patterns are emerging with progressing enactment as the data basis grows. On the other hand, the Process Navigation (PN) system executes modelled processes. We differentiate between so-called "knowledge-intensive" and "automation processes". In Section 2 we will have a closer look at this differentiation. We will justify why the application of different modelling concepts, the declarative modelling approach for knowledge-intensive processes and the imperative modelling approach for automation processes respectively, is suitable. Both types of process models can be executed by the PN system. For supporting both modelled and unmodelled business processes, the two components PN and PO must be integrated. Hence, PO makes use of collected information and generates executable process models. These models can finally be enacted by the PN system.

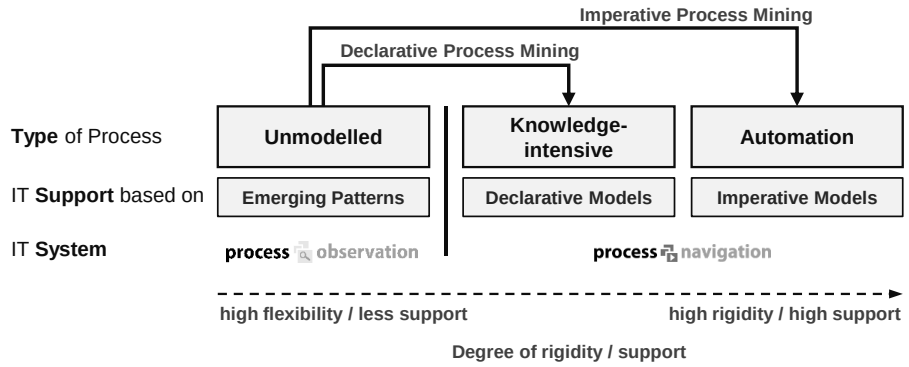

Fig. 1. Different components and conceptual structure of the approach

This paper is structured as follows: Section 2 outlines the basic principles of the execution component for modelled processes. Section 3 guides through the ideas behind the observation component for firstly unmodelled processes. Section 4 shows how the two components are integrated. Section 5 outlines the related work on flexible process execution and process mining. Finally, section 6 concludes the article, goes into the current limitations of the approach and provides an outlook on planned future work.

2 Navigating through Modelled Business Processes

Generally, IT support for business processes requires a compromise between control and flexibility [3]. Current solutions for executing modelled processes primarily focus on control. They support processes with little human participation, predetermined paths and predictable choices that focus on orchestrating services and applications [4].

We call this type of processes the “automation processes”. They are well understood and highly evolved solutions exist on the market. “Knowledge-intensive processes”, on the contrary, are driven by human participation, often contain unforeseen paths and mostly depend on human decisions. The goal of the PN system is to execute this type of business processes models.

2.1 Declarative Process Modelling

Following the terminology of programming languages, there are two paradigms of describing business process models: the imperative and the declarative style [6]. The imperative way corresponds to imperative or procedural programming where every possible path must be foreseen at design time and encoded explicitly. If a path is missing then it is considered not allowed. Classic approaches like the BPEL [5] or BPMN [4] follow the imperative style and are therefore limited to the structured type of processes. In declarative modelling, on the other hand, process models specify the possible ordering of events implicitly by constraints instead of explicitly specifying all the allowed sequences of tasks. As a result, Process Navigation relies on a declarative representation for supporting knowledge-intensive business processes.

2.2 Cross-Perspective Modelling

Declarative modelling is based on constraints that relate events of the process and exclude or not recommend certain correlations. Both constraints and events must be able to involve all the perspectives of a business process like, e.g., incorporated data, agents performing the work and utilized tools [7]. On this way it becomes possible to express realistic correlations like, e.g., the actual performing agent of a step affecting the type of data used in another step [8].

2.3 Different Modalities and Explanation

A business process usually consists of several “facets” like, e.g., a legal framework (mandatory, “must”) and best practice (recommended but facultative, “should”). Classic approaches like the BPMN only allow for describing one of these facets per model. Combining both of them in one model greatly enhances its documentary character and allows for a BPM system to act more flexibly. An action that, e.g., is contrary to best practice but conforms to the legal framework is offered but marked as not recommended. The BPM system may even explain why the action is not recommended by tracing it back to the process model.

2.4 Implementation of the Declarative Execution Core

The PN engine interprets declarative process models and recommends tasks to participants and manages process data. Therefore, it has to find next feasible actions based on the process constraints (model) and the already performed actions (log). Additionally, feasible actions need to be categorised into feasible but not recommended and recommended actions. This task is represented as a planning problem and solved by the search-based optimisation framework JBoss Drools Planner. Based on the current event log, it generates all feasible next actions within the boundaries of the process's hard constraints (e.g. legal restrictions) and scores these actions on the basis of the soft constraints (e.g. best practice). Each time an action violates a soft constraint this violation is serves for explaining to participants why the action is not recommended. The details of this implementation can be found in [9].

2.5 Imperative Execution Core and Adaptability

As mentioned above, knowledge-intensive processes do not replace automation processes. Situations remain where an imperative style of description is best suited. To come up with this situation, the PN consists of two distinct execution cores for each paradigm on a common foundation layer. Both automation and knowledge-intensive processes may be executed by the same system and many common aspects like persistence, transaction management and management of process data are shared.

3 Observing and Analysing Unmodelled Business Processes

Even though Process Navigation provides more flexibility to participants by supporting knowledge-intensive processes, the execution of business processes is still based on a predefined process model. In situations where no model can be foreseen the process must be performed without support. Comprehensive process management requires methods to overcome the separation of modelling and execution phase by applying observation and analysis methods of executed processes [10].

3.1 Observing Process Execution

In order to support unmodelled situations, the "actually" performed process needs to be recorded. Process Observation (PO) [11] provides a solution where participants record, i.e., "digitize", what they are currently doing, i.e., they provide information about the process they are performing. By providing information about the process steps as well as incorporated data objects, the system accumulates execution information that can be used to automatically generate process models and dynamic guidance feedback for future process execution [11, 14]. Additionally, it is desirable that the observation of unmodelled processes is already supported by an IT system. We iden-

tify two different types of support mechanisms: structural and behavioural support functionality.

Structural Support during Observation. Structural support mechanisms depend on process "skeletons" which consist solely of process steps. These serve as static guidelines, i.e., independent of the users' behaviour. By starting a special process, these mandatory steps are displayed to the user. This way, participants can leverage "templates" and additionally complete process information with dynamically occurring missing steps. Structural support mechanisms are implemented by adopting concepts of Adaptive Case Management (ACM) systems [12, 13].

Behavioral Support during Observation. Behavioural support functionality makes use of the recorded process execution information. PO discovers workflow patterns that provide guidelines through the process. As the data basis grows with progressing enactment, quantity and quality of discovered workflow patterns will dynamically change. We adapted association rule mining to analyse process execution logs [18]. The resulting association rules are used for guiding process participants through process execution. Therefore, the collected process execution information of the PO is periodically transformed to an input dataset for association rule mining. The transformation algorithm is described in [18]. Subsequently, the Apriori algorithm is applied to this dataset. The algorithm extracts a set of association rules. Rule (1), e.g., claims that every time process step B has been performed after step A, process step C followed (numbers represent time steps).

$$A(0) \wedge B(1) \rightarrow C(2) \tag{1}$$

The PO system manages currently extracted association rules. If a users' behaviour satisfies the left-hand side of a rule, i.e., in this case a user performed step B after step A, the right-hand side of the rule is recommended, i.e., to continue by process step C.

3.2 Evolution of Imperative Process Models

Although, the PO system supports users by structural and behavioural guidelines, the development of complete process models, e.g., for documentation purposes or workflow management system (WfMS) deployment, remains the main goal. Therefore, the PO system extracts knowledge from event logs using process mining [15] techniques. In cases where the recorded information represents a structured process, where every possible path can be described clearly, the generation of an imperative process model, e.g., a BPMN model can be initiated. There are several well-known imperative process mining techniques that can be used to generate this kind of model [15]. Since the PN system provides an engine component for imperative modelling languages like BPMN [9], extracted models can deployed and executed by the navigation system. The PO system allows for the evolution of complete process models that form the basis for future process execution with the help of a WfMS.

3.3 Extraction of Declarative Process Models

Imperative process mining techniques construct models explicitly encoding all possible behaviours [15]. In contrast to imperative modelling, declarative models concentrate on describing what has to be done and the exact step-by-step execution order is not directly prescribed. There are several process mining approaches that are discovering declarative constraints. The approach of [16] is included in the PO system and enables the extraction of declarative process models that can be executed by the declarative engine component of the PN system [9]. As the declarative models always consider all possible solutions, the number of paths through the model can become incredibly large [14]. This is why guidance through a flexible process model is necessary [17]. Therefore, best-practice workflow patterns serve as guidelines through the process. The combination of declarative guardrails and best practice guidelines finally forms an all-embracing input for powerful execution support through the application of the PN system.

4 Integrating Process Observation and Navigation

For seamlessly supporting both modelled and unmodelled business processes, PN and PO must be integrated. The first task is to find a common data model for representing the entities of process models (e.g. processes, resources, data objects) and their logs (e.g. events, projects, values). Processes may be represented as BPMN or DPML (declarative process modelling language) process models and may consist of subprocesses. Further model entities are the resources that perform process steps and data objects that may be produced or consumed. When a process is executed an instance is created. Process steps may be activated, i.e., assigned to potential performers and completed by an actual performer. When a step is completed values of data objects may be consumed and/or produced. The PN component writes execution logs, i.e., events when executing processes and may read reconstructed process models for executing them. The PO component also writes events when observing unmodelled processes and writes models after analysing event logs.

4.1 Specialization of Process Steps

One way of integrating navigation and observation is the refinement of a certain process step that has been modeled as an atomic step in the first place. The participant selects the step and chooses to let his/her work to be observed. When the originally atomic step - which is now a composite process - is finished, PO analyses the occurred events and generates a process model reflecting the recorded behavior. Here, it is necessary to choose a suitable mining method: less-structured work should be analyzed by declarative process mining while structured routine work should be analyzed by imperative mining methods. The resulting model of the composite process may now be embedded into the surrounding process model.

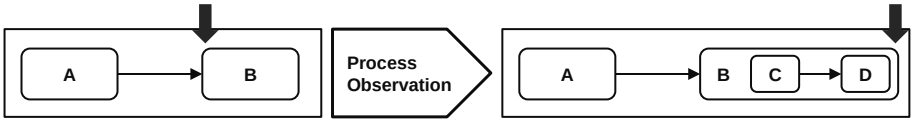

Fig. 2. Specialization of process steps by the use of Process Observation

Consider the example from Figure 2. At the beginning, a process participant is guided through a predefined process model, i.e., the workflow from step A to B, by the process navigation system. Being on the brink of performing step B, the user, who is an expert in his field, has the opinion that the abstract description of process step B could be refined. Future process performers should benefit from a more detailed description of the work to be done. This is why the user starts the PO interface where he has the possibility to record his activities, in this case two atomic process steps C and D, and preserve them for subsequent analysis. Finally, the user marks process step B as completed and returns to the navigation interface where he continues the predefined workflow. The above scenario covers a situation where existing models could be refined by expert staff while performing the process. The described functionality becomes even more useful, when we consider the application in the field. Here, process participants frequently find solutions to problems on their own. This knowledge should be preserved for future cases.

4.2 Executing Modelled Parts in Free Situations

Until now the integration of PO and PN manifested in the specialization or extension of already predefined models by observing the actual execution by the use of PO. Therefore, the PN system invokes the PO system when needed. However, there are also cases where predefined model parts can be used in free situations where process execution is completely carried out only with support of PO, e.g., the observation of whole processes that have never been modelled before. Here, users dynamically instantiate processes from predefined templates, i.e., skeletons of process models without any control flow information, and add newly occurring processes. In case that a sub-process has already been modelled before, the available process model can be executed with support of PN. Hence, the PO system invokes the PN system delivering the process to be performed. On this way, process participants can leverage a full WfMS in case of a predefined process model.

5 Related Work

The most recent approach in the field of declarative process execution is the Declare framework [19]. It is based on linear temporal logic (LTL) and therefore allows for relating process steps by temporal and existential constraints. These constraints may not contain statements on data, agents or tools. The only way of relating the temporal

order of steps to these perspectives is to make the constraints depend on certain conditions. Such a conditional constraint only applies if its condition evaluates to true. Though a condition could then contain statements on data, agents and tools, the actual constraint remains limited to temporal order and existence of steps. The other perspectives cannot be constrained, which reduces the expressivity of the supported process modelling languages. For execution, the LTL formulae of a process are transformed into a finite state automaton which will then accept every trace of events that complies with the formulae. In order to reach a technically feasible size of the automaton, only the completion of a step is considered. Though a distinction between optional and mandatory constraints is made in the theoretical preliminaries, distinct modalities are not supported because only one automaton is generated for the mandatory formulae. Both the LTL formulae and the automaton must be transformed and reduced for necessary optimization reasons. Due to that, it becomes impossible to draw a connection between the automaton's transitions and the originally modelled constraints. Therefore, Declare cannot support traceability during execution as the proposed actions cannot be explained. In spite of the simplifications and reductions, the LTL-based implementation of Declare suffers from scalability issues [20]. Process models of realistic size lead to large automata which have to be generated completely before execution. There are several approaches that are very similar to Declare. In the work of Sadiq et al. [21] and also in the work of Wainer et al. [22], temporal constraints like, e.g., serial, order and fork are used to relate steps. As for Declare, these constraints may neither depend on nor influence perspectives like data, agents or tools and modalities are not supported either.

Adaptive Case Management (ACM) reflects a more flexible approach to supporting work [12, 13]. Instead of predefining every possible process step or path, ACM systems allow participants to dynamically instantiate processes from templates as well as newly occurring processes when needed. There are already mature implementations of ACM, e.g., [17]. However, existing ACM approaches lack the use of recorded information for guidance feedback and process model evolvement and the integration with WfMS. A further process flexibility approach is the ADEPT framework [26] that enables participants to dynamically change process definitions at run time. However, the approach is still based on an imperative prescription of process models that is often not suitable to describe less-structured processes. The work at hand provides an approach to combine both worlds of process support. It relies on the recording, i.e., logging, of actually performed processes and the subsequent analysis of the accumulated execution data. We already introduced an approach for manually generating process execution data in [11]. However, this solution was not operational enough, since users where not supported, e.g., by providing process templates. Through the integration of ACM-concepts, usability considerably increased. Latest pattern recognition methods offer the possibility to extract complete process models [15] that can be deployed in WfMS. Van der Aalst et al. developed techniques and applied them in the context of workflow management under the term process mining [15]. There are several algorithms that aim at generating process models automatically and focus different perspectives of process data. Many of these traditional mining algorithms are imperative approaches [15]. These methods construct imperative models explicitly

showing all possible behaviours. Other ways to mine for process models are declarative approaches. There are several declarative discovery algorithms like [16, 23].

In cases where no complete process model could be extracted, workflow pattern mining methods can be used to find unknown coherencies in process logs. Representatives are sequence [24] or episode mining [25] that extract frequently occurring fragments of processes. However, these methods are limited to the extraction of rules considering the execution order of processes. Other types of process information, like incorporated data or agents are neglected.

6 Conclusion, Limitations and Outlook

This contribution demonstrates how to support the full range from highly controlled to fully flexible processes by integrating Process Navigation and Process Observation. Previously modelled business process parts (usually office work) are executed by the Process Navigation engine while unmodelled process parts (usually field work) are supported and reconstructed by the Process Observation system. A drawback of declarative models is that rule-based descriptions of processes generally are known to suffer from understandability issues [6]. One way of addressing this problem is to continuously simulate the execution of a process model. Therefore, a further objective is to develop a framework for the stepwise simulation of declarative process models so that their behaviour may be completely understood. Discovering workflow patterns using the Apriori algorithm may result in a high number of constraints. Many of them are trivial like, e.g., the fact that the performer of a process step is always a person. A future task is to reduce the number of constraints by identifying the trivial one.

Acknowledgement

The presented work is developed and used in the project “Kompetenzzentrum fuer praktisches Prozess- und Qualitätsmanagement”, which is funded by “Europäischer Fonds für regionale Entwicklung (EFRE)”.

References

1. Muehlen, M., Ho, D.T.: Risk Management in the BPM Lifecycle. BPM 2005 Workshops. pp. 454–466. Springer-Verlag Berlin Heidelberg (2006).
2. Zairi, M.: Business Process Management: a Boundaryless Approach to Modern Competitiveness. Business Process Management Journal. 3, 64–80 (1997).
3. Pešić, M., Schonenberg, H., Van der Aalst, W.M.P.: Declarative Workflow. In: ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., and Russell, N. (eds.) Modern Business Process Automation. pp. 175–201. Springer (2010).
4. Object Management Group Inc.: Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0, (2011).

5. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services - Version 1.1, (2003).
6. Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. Enterprise, Business-Process and Information Systems Modeling (10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009). pp. 353–366. Springer, Amsterdam, The Netherlands (2009).
7. Jablonski, S., Bußler, C.: Workflow Management: Modeling Concepts, Architecture and Implementation. Thomson, London (1996).
8. Igler, M., Faerber, M., Zeising, M., Jablonski, S.: Modeling and Planning Collaboration in Process Management Systems using Organizational Constraints. 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing. pp. 1–10. IEEE, Chicago, IL, USA (2010).
9. Zeising, M., Schönig, S., Jablonski, S.: Improving Collaborative Business Process Execution by Traceability and Expressiveness. 8th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing. pp. 435–442. IEEE Computer Society, Pittsburgh, PA, US (2012).
10. Schönig, S., Seitz, M., Piesche, C., Zeising, M., Jablonski, S.: Process Observation as Support for Evolutionary Process Engineering. International Journal on Advances in Systems and Measurements. 5, 188–202 (2012).
11. Schönig, S., Günther, C., Jablonski, S.: Process Discovery and Guidance Applications of Manually Generated Logs. 7th International Conference on Internet Monitoring and Protection (ICIMP 2012). pp. 61–67 (2012).
12. Swenson, K.: Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize The Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press (2010).
13. Fischer, L., Swenson, K., Palmer, N., Silver, B.: Taming the Unpredictable: Real World Adaptive Case Management: Case Studies and Practical Guidance. Future Strategies, Inc. (2011).
14. Günther, C., Schönig, S., Jablonski, S.: Dynamic Guidance Enhancement in Workflow Management Systems. Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012). pp. 1717–1719. ACM Press, New York, NY, USA (2012).
15. Van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011).
16. Schönig, S., Günther, C., Zeising, M., Jablonski, S.: Discovering Cross-Perspective Semantic Definitions from Process Execution Logs. 2nd International Conference on Business Intelligence and Technology. pp. 1–7. , Nice, FR (2012).
17. Kurz, M., Herrmann, C.: Adaptive Case Management – Anwendung des Business Process Management 2.0-Konzepts auf wissensintensive schwach strukturierte Geschäftsprozesse. Dienstorientierte IT-Systeme für Geschäftsprozesse, Bamberg, (2011).
18. Schönig, S., Zeising, M., Jablonski, S.: Adapting Association Rule Mining to Discover Patterns of Collaboration in Process Logs. 8th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing. pp. 531–534. IEEE Computer Society (2012).
19. Pešić, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users, (2006).
20. Westergaard, M.: Better Algorithms for Analyzing and Enacting Declarative Workflow Languages Using LTL. BPM 2011. pp. 83–98. Springer Berlin Heidelberg (2011).
21. Sadiq, S., Orlowska, M., Sadiq, W.: Specification and Validation of Process Constraints for Flexible Workflows. Information Systems. 30, 349–378 (2005).

22. Wainer, J., Bezerra, F.: Constraint-based Flexible Workflows. In: Favela, J. and Decouchant, D. (eds.) Groupware: Design, Implementation and Use. pp. 151–158. Springer, Autrans, FR (2003).
23. Chesani, F., Lamma, E., Mello, P., Montalo, M., Riguzzi, F., Storari, S.: Exploiting Inductive Logic Programming Techniques for Declarative Process Mining. Transactions on Petri Nets and Other Models of Concurrency II. 5460, 278–295 (2009).
24. Srikant, R., Agrawal, E.: Mining Sequential Patterns: Generalization and Performance Improvements. 5th International Conference on Extending Database Technology (EDBT '96). pp. 3–17 (1996).
25. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery. 1, 259–289 (1997).
26. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science-Research and Development 23.2: 81-97 (2009).