A Framework for Interactive Exception Management of Long-term Human-Involved Business Processes

Johannes Kretzschmar and Clemens Beckstein

Institute for Computer Science
Friedrich Schiller University Jena
E-Mail: {johannes.kretzschmar, clemens.beckstein}@uni-jena.de

Abstract While contemporary business modelling languages offer the possibility of describing processes on an abstract operative level there is no computer assisted support in monitoring and exception handling of such workflows. In this paper we discuss an approach to determine a formal representation of workflow execution states by applying methods of artificial intelligence (AI) planning. The hereby introduced framework *BPdoc* generates a questionnaires for human workflow participants based on semantically enriched business models and transforms the observations into a sufficient propositional state representation. The framework is triggered externally in exceptional cases and can anticipate future impacts on process soundness and correctness. By this, it serves as a foundation for further failure handling by a business administrator or by computer assisted methods.

Keywords: BPM, BAM, failure detection, exception management, AI planning

1 Introduction

Nowadays, Information Technology is an integral part of Business Process Management (BPM). Various formal languages and process engines allow the modelling and execution of complex workflows. Especially the ability of automatically monitoring process execution and visualizing comprehensive runtime information is an important benefit for upper management and process administration. The data gathered by business activity monitoring (BAM) services are essential for the optimization of workflow descriptions during the BPM life-cycle [1]. BAM provides useful information about bottlenecks or unreliable actions by evaluating triggers, counters, metrics or various performance indicators. The monitoring rely on data provided by a BP engine executing a workflow instance. Thereby BAM only works for processes with a precise executable description.
Contemporary workflow modelling languages like the Business Process Model and Notation (BPMN 2.0) [2] offer rich vocabulary to describe processes on a abstract operative level. Languages of this type are necessary for modelling processes in highly dynamic environments with human participation and nested,

complex interactions. The corresponding workflow models are generally meant for formalization, visualization and optimization. They are not executable by a BP engine and cannot be handled by BAM services for this reason.
In this paper, we introduce an approach to provide BAM-like services for such abstract workflow descriptions. The framework named *BPdoc* is a semi-automatic computer assistance for determining a current execution state, identifying failures and estimating the impact on future business activities as well as process goals. *BPdoc* operations are primarily based on methods and techniques of artificial intelligence (AI) planning.

AI Planning for Process Management

AI planning techniques are already applied to the field of BPM for evaluating, assisted developing and repairing process models. This development was primarily driven by processes as service-choreographies and related semantic service description standards [3]. But the high complexity of planning algorithms limits the application to very specific use-cases. In classical planning, there are assumptions of the domain models concerning time, execution, observability and influence aspects. These assumptions guarantee feasible planning algorithms, but in most cases there is a huge gap of expressiveness between such domain models and common workflow applications.
AI planning is based on a formal descriptive domain model [4]. This domain model provides the vocabulary to describe a state in a specific domain. This can be done, for example, by a set of propositions. A proposition is an element of a state-representing set, if the corresponding property is observable. Another part of a planning domain is a set of actions, which might be applied to states. Every action is annotated by preconditions and effects. A precondition controls, whether an action is applicable in a state and the effects describe the impact. With this information, an algorithm can calculate a new state by executing a set of actions in a given state. Therefore, the applicability, overall effects and goal reachability of processes can be computed by using methods of AI planning.
The research topic of using AI planning techniques in the field of BPM is not the focus of this paper, because we are not using the capability of planning for generating workflows from scratch. We are concentrating on the two methods from planning for applying actions (soundness) and checking states for propositions (correctness). These are essential parts of a planning algorithm for unfolding the search-space and evaluating the correctness of a solution. By leaving the costly planning algorithm aside, we also may narrow the mentioned gap and cover a wider range of workflow applications and planning domain models.

2 The BPdoc Framework for Exception Management

In this paper, we are focussing on workflows with a high abstraction level. This type of workflows is characterized by a long-lasting runtime, involving human participation, highly dynamic domain and they rely on real world resources. There exists no proper runtime monitoring for these kind of processes, because it is impractical to describe such processes on an executable level. Not to speak of the problem to install sensor technology observing the real world domain and define proper triggers and performance parameters.
We propose the framework *BPdoc*, which is able to determine a representation of the current workflow execution. This approach relies on a given semantically annotated process model, a domain description and a proposition-based representation of the state where the process was deployed. Besides the preconditions and effects, every action is annotated with a specific task associate. As shown in figure 1, *BPdoc* consists of a controller and an evaluator. The controller can generate, send, receive and process messages to and from every process participant.
For deriving a current state representation, the *BPdoc*-controller has to retrieve the state of real world workflow execution first. Each process participant sends a status messages to *BPdoc* after accomplishing a task successfully. After every message, the controller uses the *BPdoc*-evaluator to calculate an ad hoc proposition-set-representation by applying the effects of the corresponding operator from the business model. Therefore, *BPdoc* always holds a representation of the current state of workflow exeution. This representation is delayed in reference to the busines environment state, because there are only messages after every successfully accomplished task and participants may continue after sending their status messages.
The framework does not rely on continuous runtime-update messages. This approach also works by asking all participants at once, for the tasks they are currently performing and determining all the finished tasks from the process model. In cases of unordered or parallel tasks in plans, this information may be incomplete and *BPdoc* has to ask further questions. This method is, compared to the runtime-update method mentioned before, hereby more efficient, because the framework is only supposed to assist in exceptional cases and is no runtime monitoring tool.

3 Exception and Failure Detection

BPdoc gets triggered by participants or administrators in case of an unexpected event. By default, the process model and the derived propositional state only holds information about successfully completed tasks. So, the main difficulty is the formalisation of real world observations into the set-theoretic state representation of the domain model. Due the abstract and non-executable workflow models we are looking at, an automatic translation is only possible in few particular cases. To overcome this problem, *BPdoc* tries to gather information from the human process participants, who are inherent process observers on a small scope, too. But the capability of observation and autonomous failure detection counterweights a lack of formalization. Not every process participant can be assumed to be a formal knowledge engineer. We solve this problem by generating simple questions, where formal propositions according to the (semantic) process model are checked. These questions are answered in `YES` / `NO`-responses by process participants. This binary pattern correspondents to the propositional set state representation, as mentioned in section 1. This schema enables *BPdoc* to be triggered and work in two main exception cases: the non-applicability of actions and unfinished or failed tasks.

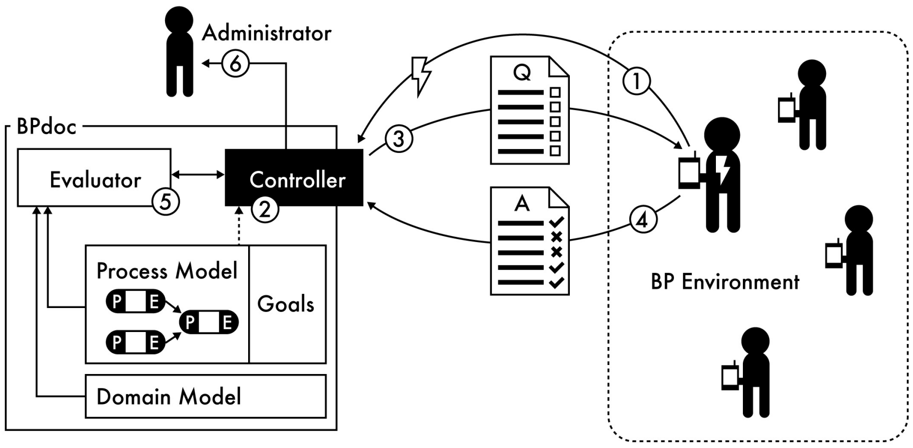

Figure 1. The interaction of *BPdoc* and the process environment

Non-Applicability Exception

This type of exception occurs, if a participants realizes that an assigned task cannot be started due missing requirements and manually triggers the *BPdoc*-controller (see figure 1 (1)). In this case, the framework has to determine the apparent discrepancy of the current state and the required preconditions of the task (2). This can by done by generating a set of questions concerning the propositions of the precondition (3). The participant has to verify these propositions by answering the question (4). By transforming the response into a virtual effect-set and applying these propositions, *BPdoc* generates an updated state representation, which reflect the current state of execution as observed by the process participant. Further *BPdoc* can test the remaining process for soundness and correctness (5) by applying outstanding workflow actions to the actual state. With help of *BPdoc*, the administrator gets information about earlier actions (6), which were not fulfilled as presumed and led to missing preconditions, an actual state representation and possible future flaws concerning the process execution or goal reachability.

Unfinished Task Exception

This type of exception arises, if a task fails in progress and the accountable participant triggers the *BPdoc*-controller. The execution semantics of state-of-the-art workflow models as well as classical planning always assume that a state remains after executing failed actions. But especially in the field of abstract human-involved workflows, activities often consist of multiple complex tasks. In these cases we have to assume, that even irregular performed actions may effect the state in an unforeseen way.
BPdoc tries to identify these side effetcs by generating questions based on the precondition and effect set of the failed operator. It is necessary to figure out, which propositions of the precondition are still existent and which effects have already occurred. The questions are generated and asked according to the non-applicability-exceptions and lead to an updated representation of the execution state. Similar to the first case, the administrator gets information about the failed execution of an action and the consequences for the process execution.

4 Conclusion

We showed that BAM-like services are also possible in processes with abstract and operative descriptions without any dedicated monitoring BP engine. The proposed framework *BPdoc* requires a semantic description of process elements and appropriate observers of the real world environment. A simple questionnaire is automatically producible and raises utilizable information corresponding a planning domain representation. Once triggered, *BPdoc* is able to determine a formal representation of the current workflow execution state and to infer prospective effects on a process. By this, the framework delivers practical information, which are essential for process repair by either a workflow modeller or an AI planner.

Our approach reveals a strong link between a domain model and failure model of processes: The more propositions a domain model holds, the finer grained is a potential detectable failure model. The *BPdoc*-framework fails to handle a specific exception, if the propositions of the domain model do not suffice to generate questions, whose answers determine the situation. So, a triggered controller but no significant change in state of execution may imply an insufficient domain model and delivers a positive refactoring impulse for the process modeller. By this, *BPdoc* may be a useful tool in the analysis and modelling part of every BPM-lifecycle.

5 Future Work

In this approach we have assumed workflow models with an control flow corresponding to a partial order plan. Modern process description languages like BPMN enable much more constructs, like loops, OR and XOR gateways. We need to stretch the plan as well as the domain model to enhance *BPdoc* handling more comprehensive workflows. Our main research in this direction is the rule language for describing aspects of the process domain. An adequate language would enable a modeller to keep the set of propositions small and would support *BPdoc* in generating efficient specific questions.
Until now, we have focused on the fundamental possibility of translating an dialogue with human process participants into a corresponding domain model. There should be a dedicated research about the usability of such a system and how it could be improved. An effective and efficient communication should be as easy and user-friendly as possible, without loosing its expressiveness.

References

1. McCoy, D., Schulte, R., Buytendijk, F., Rayner, N. and Tiedrich, A., *Business Activity Monitoring: The Promise and Reality.* Gartner, Gartner's Marketing Knowledge and Technology Commentary COM-13-9992, 2001.
2. *BPMN 2.0 specification.* `http://www.omg.org/spec/BPMN/2.0/PDF/`, 2011 (accessed March, 12th 2015)
3. Rao, J.,Su, X., *A Survey of Automated Web Service Composition Methods.* In Proceedings of 1st international workshop semantic web services and web process composition (pp. 43–54), 2004
4. Ghallab, M., Nau, D. and Traverso, P., *Automated Planning, theory and practice.* Morgan Kaufmann, 2004