Towards Efficiently Running Workflow Variants by Automated Extraction of Business Rule Conditions

Markus Döhring
SAP Research Darmstadt
Bleichstraße 8
64283 Darmstadt, Germany
markus.doehring@sap.com

Christo Klopper
SAP Deutschland
Hasso-Plattner-Ring 7
69190 Walldorf, Germany
christo.klopper@sap.com

Birgit Zimmermann
SAP Research Darmstadt
Bleichstraße 8
64283 Darmstadt, Germany
birgit.zimmermann@sap.com

ABSTRACT

Efficient workflow variant management is becoming crucial especially for enterprises with a large process landscape. Our research fosters the combination of business rules for adapting reference workflows at runtime and tailoring them to many different situations. A main goal is to optimize the performance of workflow instances w.r.t. different aspects, e.g., branching decisions, throughput time or compliance.

Having a data mining procedure at hand which can automatically extract potentially useful conditions from execution logs to create new variants is therefore a very significant benefit. The extracted conditions could be conveniently reused within the business rules of our framework, which can handle the deviations at runtime for those special situations. However, most existing data-mining techniques do not describe a continuous mining pipeline how to get from workflow logs to problematic context conditions for new variant creation or are difficult for business people to interpret.

Therefore we present an integrated rule mining methodology, starting with the semi-automatic discovery of "hot spots" within workflow instance logs. Then, data variables of instances related to these hot-spots are translated into a data mining classification problem. Other than related approaches, we employ a fuzzy rule learning algorithm, yielding easily interpretable and reusable conditions for variants. We also provide first insights from a case study at a consulting company and corresponding open research challenges.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow Management*; D.2.2 [**Software Engineering**]: Design Tools and Techniques

Keywords

workflow, business rules, process mining, process performance, rule learning

23^{rd} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 31.05.2011 - 03.06.2011, Obergurgl, Austria.

1. INTRODUCTION

Workflow management systems (WfMS) are becoming an essential part of most industrial IT system landscapes [19]. For some domains, traditional WfMS have already been determined as unsuitable to cover prevalent requirements w.r.t. the flexibility of workflows [7]. In order to address the challenge of managing workflow variants (i.e. workflows with slight deviations from a "reference workflow") at design-time as well as their dynamic adaptation at runtime due to changing data contexts, we have proposed the integration of business rules containing adaptation operations on adaptive segments in reference workflows [10].

In many practical scenarios, it is unrealistic that process analysts are able to define all variants and exceptions in a workflow. Especially when a WfMS is introduced in a company, but also if workflow models are already mature, environmental changes may lead to shifts in the impact factors on process performance. A potential relief for making such blind spots in workflow execution visible is the application of process mining techniques. The goal is to find data-dependencies for weak spots in the workflows and making them available as conditions for additional business rules leading to new workflow variants. Existing work has partly addressed these issues each with a relatively isolated view on e.g. bottleneck detection or dependency mining. Results w.r.t. to an integrated "mining pipeline" for a business user are however still quite unsatisfying. For example, prevalent approaches leave the user with a mined decision tree which, as we will show, might be hard to read for real-world workflow logs. Instead, we aim at a pipeline from a workflow definition in an understandable notation over automated mining application to interpretable business (variant) rules.

Our approach is based on the general idea of rule-based workflow adaptation as described in Section 2. As a solution to the above challenges, in Section 3 we present a mining methodology which we consider promising as a suitable mining pipeline for a business user. For each of the methodology's three generic steps, concrete technologies and their wiring are explicated, especially the employment of a fuzzy mining approach for ruleset extraction. We then present first learnings from a case study on real-world workflow execution data building upon our methodology in Section 4 and summarize challenges which have to be solved to fully implement our methodology in Section 5. In Section 6 we discuss related research, before we conclude in Section 7 and state remaining issues for future work.

2. FLEXIBILIZATION OF WORKFLOWS BY ADAPTATION RULES

Our methodology for condition extraction is motivated by a general approach for workflow adaptation [10, 9]. It is considered essential to establish a basic understanding of the nature of business(variant) rules as targeted for being automatically mined. Our framework as well as the examples in this paper rely on BPMN2 [1], because its notation is a de-facto industry standard which was designed to be understandable for business users. Basically, the framework consists of three conceptual building blocks for workflow variant management and flexible workflow adaptation:

1. Adaptive Segments in BPMN2 Reference Workflows: An adaptive segment demarcates a region of a workflow which may be subject to adaptations at runtime when entering the segment. It corresponds to a block-structured part of the workflow, i.e. a subgraph which has only one incoming and one outgoing connection. In special cases, adaptive segments can also be "empty". What matters is that they correspond to valid BPMN2 workflow definitions and not to a kind of white box which is left empty for later filling. We have extended the BPMN2 metamodel to capture the special semantics of adaptive segments [9].

2. Workflow Adaptations Defined in BPMN2: The actual definition of potential adaptations which can take place at runtime have been proposed as a pattern catalogue [10] which also relies on BPMN2 notation, with the benefit that adaptation patterns are comprehensible and extensible. The catalogue contains basic adaptations like SKIP or INSERT, but also more sophisticated event- and time- related patterns, like "event-based cancel and repeat" or "validity period". Every adaptation pattern has the block-structured adaptation segment as an obligatory input parameter. As such, patterns can be conveniently nested and combined.

3. Linking Adaptations to Data Contexts by Business Rules: Business (variant) rules are used to apply suitable adaptations for different situations expressed by data context conditions. The data context can be globally valid (like a date) or workflow instance specific (like an order value). A pseudo-syntax for variant rules, where $*$ stands for 0-n repetitions, can be defined as: **ON** *entry-event* **IF** *<data-context>* **THEN APPLY** [*<pattern(segment, (parameter, value)*>*]$*$ Once the general relations of adaptive segments and potential adaptations have been established by a process analyst, the conditions could be maintained by a business user e.g., via a domain-specific language. For automatic rule extraction, in this work we therefore especially focus on the IF-part of potentially newly discovered variant rules and aim at revealing data dependencies for variants which are not a-priori known, but have significant implicit impact on the overall business performance of workflow execution.

Figure 2 exemplifies the above concepts based on a ship engine maintenance workflow fragment. The actual conduction of engine tests for a ship may depend on the harbor in which it currently resides. Due to environmental restrictions, many different harbors impose specific time constraints on ships conducting engine tests. In Hamburg for example, ships may only have 12h time, after which devices need to be reset and the tests need to be restarted. For adapting the workflow correspondingly, a generic parameterizable template is used and weaved with the segment at runtime.

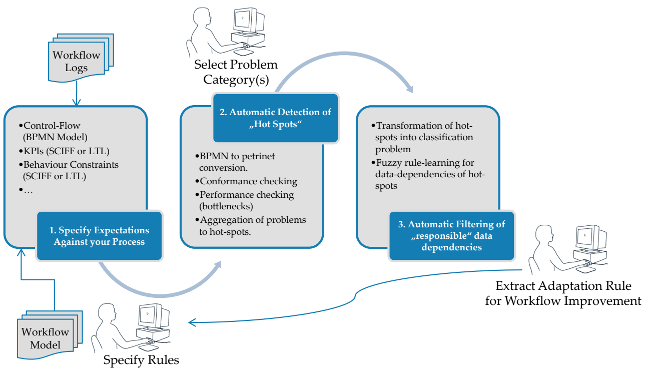

Workflow Logs
Select Problem Category(s)
•Control-Flow (BPMN Model)
•KPIs (SCIFF or LTL)
•Behaviour Constraints (SCIFF or LTL)
•…
1. Specify Expectations Against your Process
2. Automatic Detection of „Hot Spots“
•BPMN to petrinet conversion.
•Conformance checking
•Performance checking (bottlenecks)
•Aggregation of problems to hot-spots.
•Transformation of hot-spots into classification problem
•Fuzzy rule-learning for data-dependencies of hot-spots
3. Automatic Filtering of „responsible“ data dependencies
Extract Adaptation Rule for Workflow Improvement
Workflow Model
Specify Rules

Figure 1: Outline of the Rule Extraction Procedure

3. METHODOLOGY FOR VARIANT RULE CONDITION EXTRACTION

As already stated, we are interested in automatically extracting condition constraints (the "IF-part") for potentially useful workflow adaptation rules within our framework. Useful in this respect means, that the condition constraints should describe eventually problematic situations in workflow instances by means of their data context values, such that a timely adaptation of a workflow instance can eventually prevent such a situation. Our proposed methodology is illustrated in Figure 1 in a circular manner. The methodology is divided into three main phases explained in detail in the following subsections. For each phase, concrete concepts and technologies for implementing the methodology are discussed and open challenges are outlined where existing.

3.1 Formulation of Log Expectations

The first phase of our methodology consists in the definition of expectations towards a set of workflow instance logs. Correspondingly, there are two obligatory input components for the extraction pipeline: a workflow model and a sufficiently large set of workflow instance logs belonging to the model. The instance logs must contain workflow-relevant events like at least the start or finishing timestamp of particular task types and must also carry a number of data context variables[1]. Since we want to target business users with our rule extraction approach, we consider BPMN as an appropriate input format for the expected control-flow logic restricing the expected order of task executions and event occurrences in the input logs.

As an optional input, additional constraints w.r.t. workflow execution can be provided in some form of logic. These constraints may concern time-related interdependencies of events within a workflow instance log, whereas typical key performance indicators (KPIs) like throughput times can be understood as a subset of such time constraints. But also other more sophisticated circumstances which are hard to model in BPMN2 graph structures can be provided as logical constraints, as for instance that a task *A* should be executed N times after the occurrence of task B. Suitable logics to formulate such process-related constraints can for example be based on the SCIFF framework [5] or linear temporal logic (LTL) [17]. Since a regular business user may not

[1]It is hard to give generally valid recommendations on data size characteristics, but from experience reasonable mining can start from 1000 instances with about 5 context variables.

be familiar or feel comfortable with such logics, it is recommended to provide constraint templates, i.e. small chunks of logic mapped to easily parameterizable pieces of restricted natural language for constraint maintenance.

3.2 Automatic Discovery of "Hot Spots"

For the ability to apply established mining and analysis techniques on the instance logs in combination with the workflow model, it is useful to first transform the BPMN workflow definition into a pure formal representation, e.g. in terms of petri net graphs which are backed by a long trail of research and corresponding toolsets. Transformation mechanisms which are able to map a large part of BPMN constructs to petri net constructs exist [8] and can be employed within our methodology. The next phase of our methodology then consists in the automatic discovery of problematic spots in the instance logs, relating to different issues:

1. **Non-conformance to defined workflow model**: Using log-replay approaches on the petri net model as presented in [16], it can be determined whether instances behave exactly according to the underlying model or whether there are deviations. Provided the petri net has been suitably constructed, such deviations can be structurally spotted as petri net places where tokens are left over after an instance has been finished or where tokens often are missing when a transition should be fired. In most of the latter cases, a distinct transition (=BPMN task) can be "blamed" for causing the non-conformance. Places and transitions with a relatively high error-rate are kept for further analysis within our methodology.

2. **Disproportionate delays (bottlenecks)**: Similar to the above petri net log-replay techniques, the sojourn times of tokens in places and the times it takes to execute transitions can be stored [12]. Based on this computed data, it can be determined where instances on average get stuck for a disproportionate amount of time related to the average overall throughput time. The corresponding threshold values can be computed automatically if they are not explicitly formulated as KPI constraints, which is discussed below. Again, concerned places and transitions are kept for analysis.

3. **Non-conformance to execution constraints**: SCIFF or LTL constraints can be checked on the instances logs using approaches from [5] resp. [17] with respect to their violation. The employment of constraint checking allows for a very broad range of non-conformance types being checked. Three of the most important ones are:

   - The violation of KPIs by the use of time-related constraints (for example, task B has to be executed 1h after task A latest).
   - The deviation from expected routing decisions (for example if orderValue>10.000 in a sales order, always choose the "priority shipment" branch after an exclusive gateway).
   - Data- or organizational incompliance like the violation of the "four-eyes principle" for some tasks.

   In contrast to the checking mechanisms for issues (1.) and (2.), a challenge consists in the spotting of the actual source for a constraint violation. For our KPI example (B 1h after A), if B is not executed at all, it has to be decided whether *A* or *not B* or *both* are to be considered as the actual error source and kept for further analysis. Potentials lie in the partly automated mapping of constraint predicates to places or transitions in the underlying model and the consideration of "what happened first". Research is still ongoing here.

As a final step of this phase, the user is confronted with issues which have a particular degree of "severity" (e.g. exceed a predefined fraction of instances which are non-conformant) and gets the corresponding "hot-spots" based on average instance execution marked in the BPMN process model. The proper automatic accumulation and back-projection of issues to the BPMN workflow model remains an open issue. The user may then select one or several hot spots and one or several problem types for these hot-spots for further analysis by mining data dependencies as business rule conditions as described in the next subsection.

3.3 Automatic Extraction of Rules for "Hot Spot Occurrences"

For the selected hot-spots and problem types, the instance data from the workflow logs is transformed into a classification problem for machine learning algorithms. A classification problem consists of a number of cases (=workflow instances), each made up of a number of numeric or nominal data variable values (=workflow instance or task context, e.g. order value, customer priority or shipment partner) and a single class in terms of a category for a learning instance. The class can be determined in a binary manner as *problematic* or *non-problematic* from the problem types connected to the hot spots, but also the distinction of finer-granular problem classes can be considered. The variable values for a learning instance can be constructed by looking at their occurrence when an instance has reached a hotspot in the petri net. Special challenges in this conversion step concern the treatment of some control-flow constructs, as for example a loop which may cause multiple visits of a hotspot in a workflow, whereas the context variables may have changed meanwhile. Such problems and solution approaches, for instance creating a separate training instance for each loop execution, are discussed, e.g., in [15].

Having the training set for a machine learning classifier at hand, established algorithms like C4.5 decision tree [14] or rule learners [6, 11] can be applied. In fact recent research mostly favors decision trees for presenting mining results to the business user [18]. However, we have tested the C4.5 decision tree learner on a real-world dataset (see Section 4) and found its results not interpretable for the business user to draw any reasonable conclusions from it mainly due to the size and complexity of the overall decision tree. Despite ex-post global optimization heuristics in C4.5, local feature selection often leads to redundant splits in the initial decision trees. As rules can only be extracted one-by-one along paths in the decision tree [11], they are of rather less use for directly extracting conditions for use in adaptation rules that might eventually tackle the problematic situation at workflow runtime. The problem with established rule learners like RIPPER [6] in turn is that they generate *ordered* rule-lists, which means each rule in the list covers only those learning instances which are not covered by the previous rule. This characteristic makes the corresponding output rules also hard to read and interpret for an end user. Potential relief consists in the employment of a fuzzy learning approach which generates globally valid rules that have a probabilistic certainty factor to hold on the dataset or not. We are currently evaluating a novel algorithm [13] w.r.t. the suitability for being employed within our methodology, which is subject to discussion in the following section.

4. CASE-STUDY

The first feasibility study for our methodology was conducted at a large globally operating IT consulting company. In the following, we report on the input dataset, the realization of our methodology in the ProM[2] framework, and our preliminary results and findings.

4.1 Description of the Dataset

The focus of the case study is on a staffing workflow for serving customer and company-internal human resource requests for different type of IT projects. A simplified corresponding model in BPMN notation is shown in Figure 3. The first three sequential steps are creating and submitting the request and then having it validated by an authorized person. Resources can be found by three different strategies: by company-internal broadcasts, by external broadcasts to partner consulting companies or by directly contacting a potentially suitable resource. After at least one such search procedure has been triggered, different reactions can occur, namely the acceptance, rejection, withdrawal or feedback of non-availability for a particular resource. At anytime during these search procedures, an initial proposition of currently gathered resources can be made to the customer. After the request is closed, it is marked as either successfully or not staffed. The input dataset consisted of 13225 workflow instance logs each with up to 50 data context variable values attached. In this case, context variables concern for example the country a request is sent from, the concerned industry profile or the overall duration of the project.

4.2 Realizing the Methodology based on ProM

For some basic analysis techniques, we rely on functionality provided by ProM. The translation of the BPMN model into a petri net was done manually, as automated mapping approaches still generated too complex results which could make first mining and analysis efforts more difficult. The resulting petri net is shown in the upper middle of Figure 4. Black boxes indicate "silent" transitions which do not correspond to any task in the BPMN model. On the left upper side, one of the additional constraints provided by the consulting company for its staffing workflows is shown, i.e. that before or at least in parallel to an external broadcast, there should also be an internal broadcast trying to gather the required resources. The lower left window shows the evaluation results of these rules. In the right window, the petri net-based bottleneck analysis indicates an overproportional waiting time between request submission and request validation (concrete values in the figure have been changed for anonymization purposes). In the lower middle window, we see an instance marked with a conformance issue, namely that the request validation sometimes has been left out or was conducted only after another task already was executed. Combining these information types, we would identify the validation task as a "hot spot" in the process.

For our first analysis purpose however, we have concentrated on the decision whether a request has been staffed or not. Following [15], we turn the decision into a binary classification problem using a manually selected subset of context variables that have occurred while instance execution. The results are presented in the following.

[2]http://www.promtools.org/prom5/

4.3 Preliminary Results

Running a C4.5 decision tree (J48 implementation) learner with standard parameters yields a decision tree of size 757 with 644 leaves. It is quite obvious that this output type would need a considerable time to be interpreted for a business user. Leaving aside the rule learning algorithms for ordered rule lists, we instead applied the fuzzy rule induction algorithm presented in [13]. Results were very promising, for example generating the following output (some context values changed for anonymization):

```
(Remote = Y) and (ReqingSRegion = DUCKBURG) and (ReqType = Project)
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.61)
(ReqingSRegion = NA) and (StartDateFlexible = Yes) and
  (ReqingLOB = FS__Consulting) and (CustIndustry = )
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.71)
(Remote = N) and (ContractType = ) and (CustIndustry = UTILITIES) and
  (JobText = B) and (Requestor = ABC) and (StartDateFlexible = No)
    => class=Branch 4.1 { ROLE_Closed (Not Staffed)/complete } (CF = 0.53)
(Remote = ) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.73)
(Remote = Y) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.7)
(ReqingSRegion = GOTHAM_CITY) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.72)
(StartDateFlexible = No) => class=Branch 4.2 { ROLE_Staffed/complete } (CF = 0.72)
```

Manual inspection of the instances characterized e.g. by the first two rules immediately showed that they in fact constitute problematic situations in the staffing workflows. In a flexible WfMS according to Section 2, these conditions could now be reused as a condition for a variant rule with the click of a button, for example inserting addtional activities in the workflow to handele the problematic situation or not even trying specific activities because of potential waste of time.

5. OPEN CHALLENGES

For a better overview and to motivate future work in this area, the main challenges we experienced while setting up the mining pipeline are briefly recapitulated:

- A petri net conversion most useful for mining purposes has to be determined, as straight-forward mappings have problems with more advanced BPMN constructs or generate valid but overcomplex petri nets.
- The accumulation and aggregation of hot spots from the petri net-based and especially the constraint-based checking methods has to be defined in more detail. This challenge is connected to linking back hot spots to the BPMN model for further investigation.
- The conversion of hot spots to a classification problem has to be advanced w.r.t. problematic control-flow structures as for example loop or special joins.
- For the classification problem, the selection of context variables and algorithm parameters has to be made accessible for a business user. Experiments also showed that the rule output may vary significantly w.r.t. the predicates used in the rules. We have to find a way for stabilizing the rule output, e.g. by modifying the learning algorithm w.r.t. this goal and not only taking prediction accuracy into account.

6. RELATED WORK

Due to space restrictions, we do not cover the broad range of general process mining approaches in this section, but rather elaborate on selected approaches which tackle the issue of dependency- or constraint-extraction in workflow logs:

The authors of [15] present the idea of decision point mining in workflows by translating a routing decision into a classification problem for machine learning. In this work, we generalize this idea also for problem domains in workflow execution like bottlenecks or general rule compliance. In [18], a pipeline for analyzing influential factors of business process performance is presented. Some of the steps resemble that of our approach, however e.g. decision trees are used for dependency analysis. The approach is evaluated on a simulated dataset. As we have motivated, decision trees are rather unsuited for direct extraction of globally valid "hot-spot" conditions for a business user on real-world data. An approach for learning constraints for a declarative workflow model is presented in [4], however focusing on control-flow constraints and neglecting data-dependencies. In [3], related to HP's solution for business operation management, an overview on the suitability of different mining techniques for specific analysis types are discussed. Rule extraction is mentioned, but only as rules derived from decision trees which as discussed may get too complex for our purposes. The approach in [2] focuses on dependencies of service-level agreements for service compositions and analyzes reasons for SLA violations. In contrast to our approach, where dependencies are extracted from historic data, the dependencies in [2] are identified at design time for later comparison with monitoring results at runtime.

7. CONCLUSION

We motivated the need for automated extraction of condition constraints for problematic "hot spots" in workflows by the initial uncertainty of a modeler when introducing a flexible WfMS and by rapidly changing impact factors on workflow execution performance. Existing approaches for data dependency extraction have turned out not to deliver conveniently interpretable results on real-world datasets and were considered generally hard to employ for business users.

Therefore in this work we have proposed a methodology which starts from a BPMN workflow definition with a set of additional template-based constraints and transforms the workflow into a petri net for automatic hot-spot discovery according to rule-conformance, control-flow-conformance and bottleneck detection. The hot-spots in turn are transformed into a classification problem for further mining algorithms which should explain the data-dependencies characterizing the problem. One key differentiator to other approaches is the use of a fuzzy rule induction approach, which delivers globally valid and interpretable rules. Our approach especially aims at providing the corresponding conditions for reuse in adaptation rules which improve the overall workflow performance by circumventing critical situations.

However, some integration steps between the phases of our methodology, like a BPMN to petri net translation suitable for mining purposes, the aggregation of problem situations to hot-spots or the guided parameter selection for the rule mining algorithm remain subject to future work.

8. REFERENCES

[1] Business Process Model and Notation (BPMN) - Version 2.0 11-01-03, 2011.

[2] L. Bodenstaff, A. Wombacher, M. Reichert, and M. C. Jaeger. Monitoring Dependencies for SLAs: The MoDe4SLA Approach. *SCC'08*, pages 21–29, 2008.

[3] M. Castellanos, F. Casati, U. Dayal, and M.-C. Shan. A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. *DAPD*, 16(3):239–273, Nov. 2004.

[4] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, and S. Storari. *Exploiting Inductive Logic Programming Techniques for Declarative Process Mining*, pages 278–295. Springer, 2009.

[5] F. Chesani, P. Mello, M. Montali, F. Riguzzi, and S. Storari. Compliance Checking of Execution Traces to Business Rules. In *BPM'08 Workshops*, pages 129—-140, Milan, 2008. Springer.

[6] W. W. Cohen. Fast Effective Rule Induction. In *ML'95*, pages 115—-123, 1995.

[7] P. Dadam and M. Reichert. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. *CSRD*, 23(2):81–97, 2009.

[8] R. M. Dijkman, M. Dumas, and C. Ouyang. Semantics and Analysis of Business Process Models in BPMN. *IST*, 50(12):1281—-1294, 2008.

[9] M. Döhring and B. Zimmermann. vBPMN: Event-Aware Workflow Variants by Weaving BPMN2 and Business Rules. In *EMMSAD'11*, London, 2011. Springer.

[10] M. Döhring, B. Zimmermann, and L. Karg. Flexible Workflows at Design- and Runtime using BPMN2 Adaptation Patterns. In *BIS'11*, Poznan, 2011. Springer.

[11] E. Frank and I. H. Witten. Generating Accurate Rule Sets Without Global Optimization. In *ICML'98*, Madison, 1998.

[12] P. Hornix. Performance Analysis of Business Processes through Process Mining. (January), 2007.

[13] J. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *DMKD*, 19(3):293–319, Apr. 2009.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc, 1993.

[15] A. Rozinat and W. van Der Aalst. Decision mining in business processes, 2006.

[16] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *IS*, 33(1):64–95, 2008.

[17] W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process Mining and Verification of Properties. In *OTM Conferences (1)*, pages 130—-147, Agia Napa, 2005. Springer.

[18] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. *EDOC'09*, pages 141–150, 2009.

[19] P. Wolf, C., Harmon. The State of Business Process Management 2010, 2010.

IF dockyardStation==*Hamburg* **THEN APPLY** NonFailableTimedHandler(measurements, time=*12h*, handlerTask=*ResetDevices*)

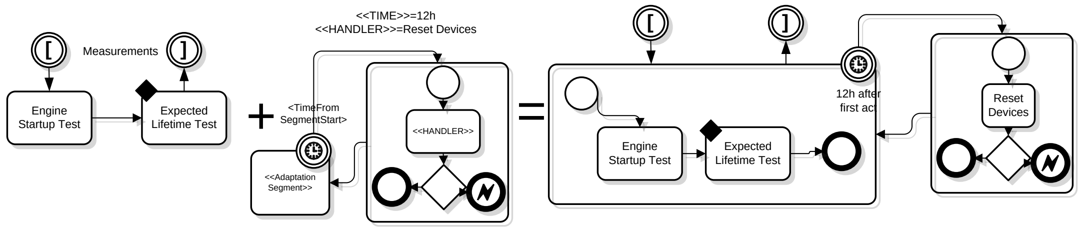


Figure 2: Example of a Rule-Based Workflow Adaptation

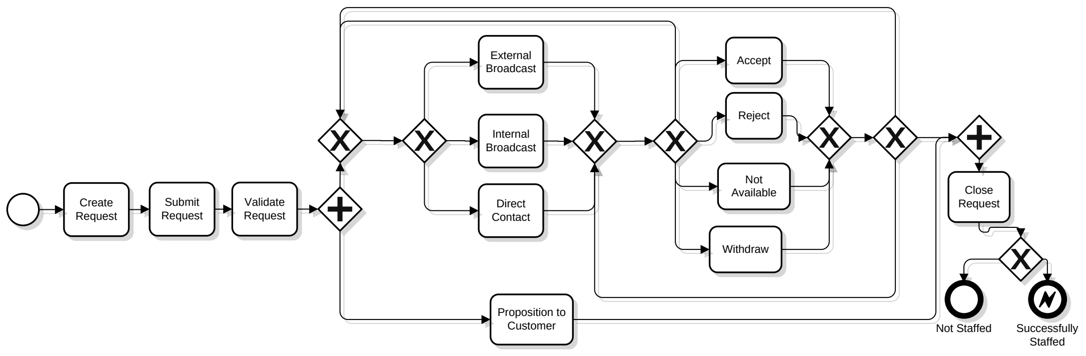
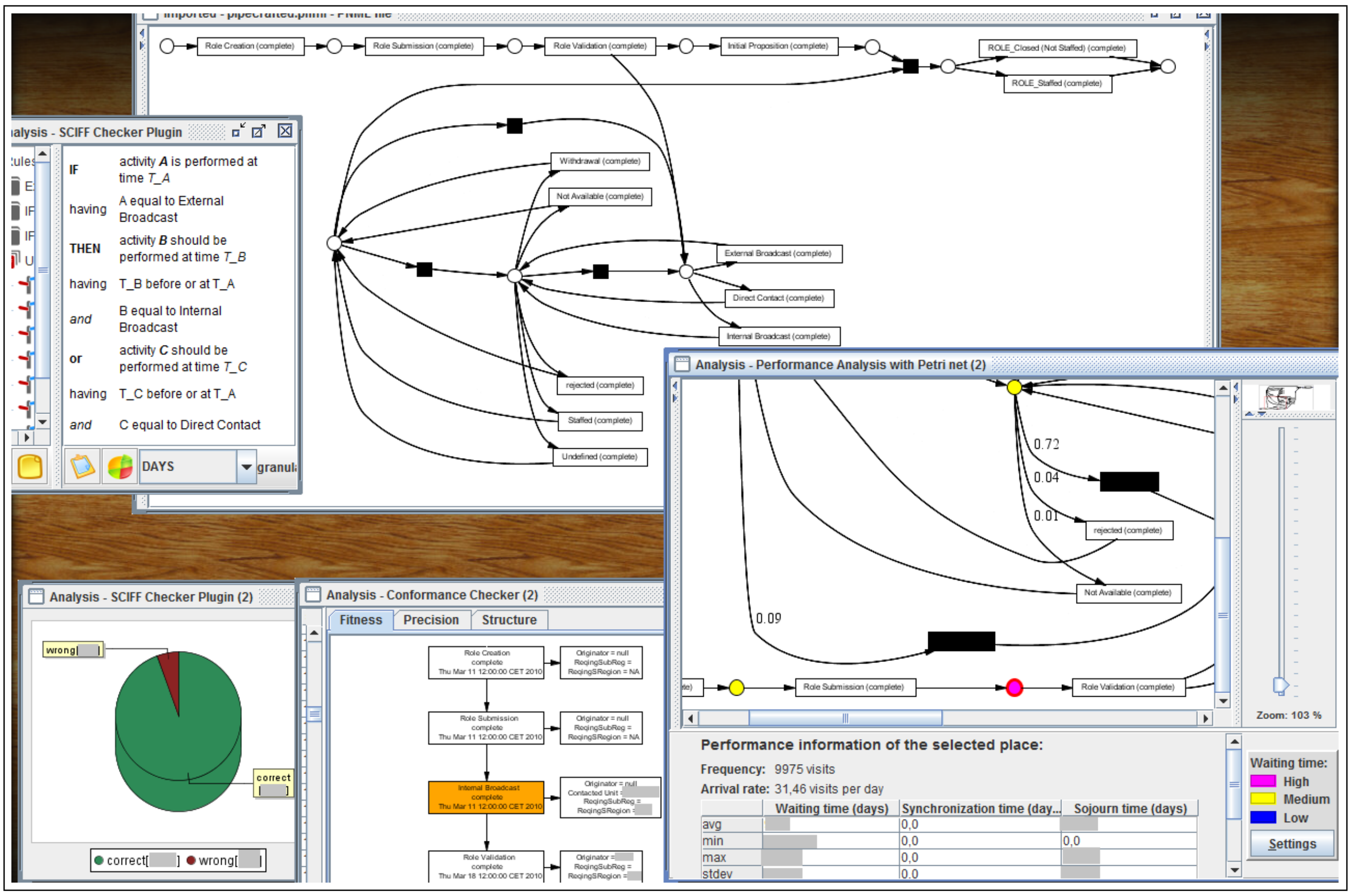


Figure 3: Staffing Workflow of a Large Globally Operating IT Consulting Company



Role Creation (complete)
Role Submission (complete)
Role Validation (complete)
Initial Proposition (complete)
ROLE_Closed (Not Staffed) (complete)
ROLE_Staffed (complete)
Withdrawal (complete)
Not Available (complete)
External Broadcast (complete)
Direct Contact (complete)
Internal Broadcast (complete)
rejected (complete)
Staffed (complete)
Undefined (complete)
Analysis - SCIFF Checker Plugin
DAYS
Analysis - SCIFF Checker Plugin (2)
wrong
correct
● correct[ ] ● wrong[ ]
Analysis - Conformance Checker (2)
Fitness
Precision
Structure
Analysis - Performance Analysis with Petri net (2)
0.72
0.04
0.01
0.09
Zoom: 103 %

IF	activity *A* is performed at time *T_A*
having	A equal to External Broadcast
THEN	activity *B* should be performed at time *T_B*
having	T_B before or at T_A
and	B equal to Internal Broadcast
or	activity *C* should be performed at time *T_C*
having	T_C before or at T_A
and	C equal to Direct Contact

Performance information of the selected place:

Frequency: 9975 visits

Arrival rate: 31,46 visits per day

	Waiting time (days)	Synchronization time (day...	Sojourn time (days)
avg		0,0	
min		0,0	0,0
max		0,0	
stdev		0,0	

Waiting time:
🟪 High
🟨 Medium
🟦 Low
Settings

Figure 4: Screenshot of ProM with Most Relevant Workflow Analyses within our Methodology