Extended Workflow Flexibility using Rule-Based Adaptation Patterns with Eventing Semantics

Markus Döhring, Birgit Zimmermann, Eicke Godehardt

{markus.doehring, birgit.zimmermann, eicke.godehardt}@sap.com

Abstract: In several industry scenarios, it is often the case that an existing reference workflow has to be adapted according to specific context factors, which might even change at runtime. The adapted workflow instances virtually constitute process variants. In order to keep the efforts for systematic variant configuration and maintenance on a manageable level, recent work has proposed the use of context-dependent adaptation rules. The involved rule-based change operations are however usually restricted to simple constructs like task deletion or insertion. This can be an obstacle if eventing paradigms for modeling reactive parts of a workflow also should be combined with context-awareness.
In this paper, we present an example machinery maintenance service delivery use-case and show how hierarchical context rules can be integrated to tailor the workflow to changing data contexts. We furthermore propose to extend existing basic change operations with an adaptation pattern catalogue that especially captures event-based adaptation semantics for workflow languages like BPMN2[1] and show how a part of our solution was prototypically implemented in jBoss Drools.

1 Introduction and Motivation

Today's process-aware enterprise information systems are facing a tough challenge: they are expected to provide mechanisms for the design, execution and monitoring of standardized business logic *and* at the same time to allow for sufficient degrees of freedom for context-dependent resp. situation-based tailoring of IT-based procedures [vPS09]. In many business domains it is often the case that a reference workflow (template) exists which sometimes needs to be adapted to a variable context. The context might be completely determinable at instantiation time or even change at runtime [HBR09a]. Such adapted workflows constitute variants, whose explicit one-by-one modeling implies the danger of significant redundancies and becomes practically infeasible if the number of possible variable-value combinations based on which a variant is configured is too high.

To address this issue, business rules have been proposed in several works to condition the execution of design- or runtime change operations to workflow templates on context variables. These change operations as presented in existing work are however often restricted to simple constructs like task deletion, insertion, movement or replacement. They lack the

[1] See http://www.omg.org/cgi-bin/doc?dtc/09-08-14

provision of immediate context-dependent reaction mechanisms for workflow languages like BPMN2 that explicitly support eventing paradigms. The proper integration of workflow and complex event processing (CEP) paradigms, which can discover business relevant events by filtering and correlating large noisy sets or streams of atomic events, is considered crucial for a highly responsive real-time enterprise architecture [Lun06, DKGZ10].

In this paper, we present first ideas for developing an extended adaptation pattern catalogue for the BPMN2 language that especially allows a modeler to realize context-dependent dynamic runtime reactions on events from within or from outside a workflow engine. Related work is discussed and our approach is distinguished against others in section 2. In section 3, an example BPMN workflow is introduced together with basic rule-based adaptation mechanisms. Section 4 presents the idea of an extended event-driven adaptation catalogue for BPMN2 as the main conceptual contribution of this work. Section 5 contains a discussion of the main implications and challenges faced by our approach. A part of our solution has been prototypically implemented in jBoss Drools, which is briefly presented in Section 6. Section 7 concludes this work.

2 Related Work

Rule-based adaptation is not the only mechanism to provide flexibility in a process-aware information system. [PSWW05] provide a taxonomy of variability mechanisms for process models and show how they can be expressed in UML, BPMN and Matlab/Simulink. One prevalent alternative to rule-based adaptation is the creation of parameterizable reference workflows, that contain all possible execution possibilities. A variant is then created by "fading out" irrelevant parts of a workflow [RV07, vdADG+08, GvJV07]. The acquisition of context variable values can be achieved by mapping them to a simple questionnaire that can be understood by end users [LVDT08].
Instead of creating a large parameterizable workflow model, the other way round is also possible. [LDKD09, SKY06, AHK05] present methods for merging single workflow models into a larger reference workflow.

One of the approaches for context-rule based change operations for variant creation most related to this work is presented in [HBR10], which also specify variant points in a workflow together with change operations (INSERT, DELETE, MOVE, MODIFY) and provide mechanisms e.g., to provide consistency of the adapted workflow when a context variable changes at runtime. [KY09] show how process templates expressed in Prolog can be materialized depending on the value of predicates. The defined adaptations can concern the control-flow as well as the data and the resource perspective of a workflow. These works however do not especially consider event-based reaction resp. exception handling mechanisms as proposed by us. The work of [LSGY09] especially focuses on how constraints can be expressed that restrict for example how an ad-hoc part of a BPMN workflow can be adapted by a user. Rule-pattern based workflow modeling is for example discussed in [KRSRS96, CCF+00]. A prototypical implementation using jBoss Drools that shows how BPEL processes can be extended with rule-based self-healing functionality for exception

handling is presented in [BGP07]. The approach has similarities to ours in terms of attaching compensation mechanisms to workflows via rules, however it is based on the extensive specification of postconditions and does not focus on eventing semantics.

To the best of our knowledge, none of the discussed work fosters the introduction of a pattern catalogue to enable context-specific event-driven adaptations to workflows. As such, existing work especially lacks guidance how to actually combine and employ their means of flexibility [DKGZ10], a gap which we will address in the following.

3 Context-Aware Process Adaptation

To illustrate our proposed rule-based workflow adaptation mechanisms, a fictitious but realistic workflow for ship engine maintenance is presented in figure 1 in BPMN2 notation. At first, the ship engine is set to maintenance mode. Depending on the type of the cooling component (water or air-cooled), either a maintenance of the cooling fans and air filters or of the cooling liquid reservoir and pump system is conducted. In parallel, if the ship resides in a dockyard for maintenance, additional tests are conducted, which might require an engine startup that in turn needs approval from the dockyard as due to emission restrictions, ships must not arbitrarily start their engines while docked. At the end of the process, missing or damaged engine spare parts are eventually replaced if the customer is solvent. As we will see later, it might be required that the added spare parts have to be revoked, so a corresponding compensation handler is included in the flow.

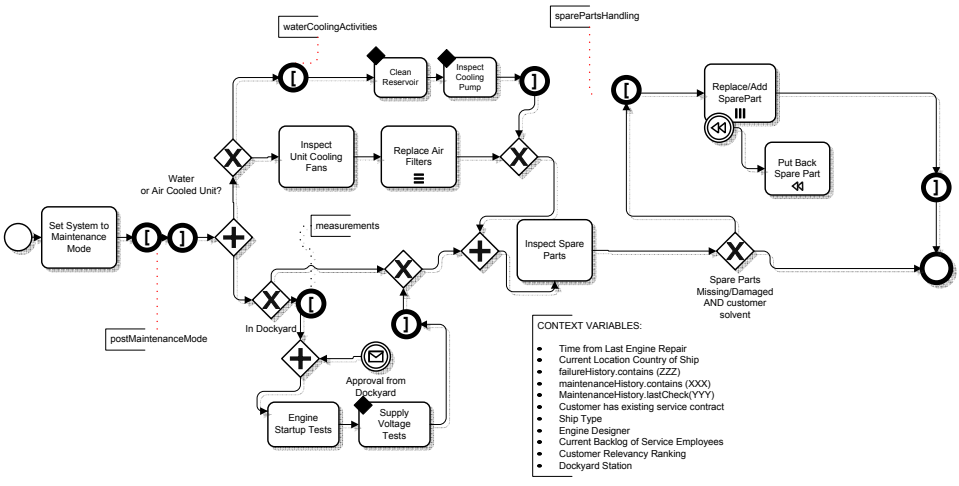

Figure 1: Example Service Delivery Workflow for Engine Maintenance

The diagram in figure 1 depicts the *normal* progress of the workflow. It however also contains the listing of some context variables, based on whose value the workflow may

deviate from its original modeling. To facilitate the support of consistency at design-time and the handling of adaptations at runtime, we build upon concepts from [HBR09a] to define regions resp. variability points in a workflow marked by circles with opening and closing square brackets. We restrict variable regions to be block-structured, i.e. there might only be one entering and one outgoing flow connection. However we do not require the whole workflow to be block-structured. It is also allowed to mark a single task as variable (e.g., for direct reference for deletion or replacement) by placing a black diamond on the task node. According to [HBR09a], explicitly modeling the variable points in workflow is more robust than using absolute references, e.g. to task IDs for region adaptations. In figure 1, we have four variable regions marked with a BPMN2 annotation on the variant start node (opening square brackets). These are namely *postMaintenanceMode*, *sparePartsHandling*, *waterCoolingActivities* and *measurements*. There are also four variable tasks: *cleanReservoir*, *inspectCoolingPump* and *supplyVoltageTest*. If a common task definition repository for rules and workflow modeling exists as stipulated in [DKGZ10], we can express parametrized hierarchical adaptation rules conditioned on the context variables as shown below that apply the common change operations *insert*, *delete*, *move* and *replace* on the workflow graph:

```
RULE #1: IF engineDesigner==DESIGNER1 DELETE cleanReservoir
RULE #1.1: IF maintenanceHistory.contains(frequencyConverterChanged)
             INSERT checkConverter WITHIN postmaintenancemode
RULE #1.2: IF maintenanceHistory.contains(oilLeakageRepair) INSERT integrityCheck WITHIN postmaintenancemode
RULE #2: IF (shipType)==passenger ship INSERT Turn Off Security Locks WITHIN postmaintenancemode
RULE #2.1: IF (existingServiceContract) AND currentBacklog<80\%
           INSERT extensiveLifetimeAnalysis WITHIN postmaintenancemode
RULE #3: IF !(existingServiceContract) DELETE ALL measurements; DELETE Replace Spare Parts
RULE #4: IF yearsFromStartup(pumpSeals)>4 OR yearsFromRepair(pumpSeals)>8
         REPLACE inspectCoolingPump<>replaceCoolingPump
```

The concept of employing hierarchical "Ripple-Down-Rules" for introducing workflow execution dynamics, where a descendant rule extends a parent rule, was amongst others presented in [ATEV06]. We can see that rule #1 states that it is unnecessary to clean the cooling liquid reservoir for a specific engine designer. For the same designer however, additional checks need to be conducted depending on the past maintenance history (rules #1.1 and #1.2). In rule #2.1 we can see that a context variable (the worker backlog in this case) can possibly change at runtime. Therefore the rules that include runtime-variable conditions in their LHS[2] are evaluated right at the point where the workflow enters a variable region that is covered by their RHS[3]. The adaptations applied to the variable region or single task however remain constant for a single flow pass, i.e. they are not readapted if a context-variable changes again during execution. That means if a flow passes a variable region entry point (defined by square bracket nodes) multiple times e.g. through a cycle, it might be the case that a variable region is executed in multiple different variants within a single workflow instance.

[2]Left-Hand-Side (LHS) means the condition part of a rule.

[3]Right-Hand-Side (RHS) means the action part of a rule.

Table 1: Event-Driven Adaptation Pattern Catalogue Draft

ID	Name	Description	Parameters
#1	UnfailableTimedHandler	Sets a boundary error event to the concerned region that is signaled according to a specified interval starting from the start of the first task within the region. If the timer expires, the region is canceled and a compensation mechanism is triggered, followed by the throwing of an escalation event.	TIME, HANDLER
#2	CompensationWithEscalation	If the timer expires, the region is canceled and a single task handler is called. If the handler succeeds, the region is re-started. If the handler fails, the whole process fails.	REACTONEVENT, COMPENSATETASK
#3	WaitForEvent	Delays the continuation within a flow until a particular event occurs. If the event does not occur within a given period of time from the start of the workflow instance, the instance terminates with an error.	WAITEVENT, DURATION
#4	RestartEvent	If the specified event occurs, the concerned variable region or task is canceled and immediately restarted	RESTARTEVENT

4 Extension to Event-Driven Adaptation Patterns

Up to now, we only dispose of a conceptual facility to specify *what happens next (or not)* in the workflow instance. What is missing, and this point is examined in only few of the related work (see Section 2) on dynamic rules based adaptation, is how to deal with running task instances when sudden events occur that in turn need immediate context-dependent reaction. These events may come from within the workflow engine constituting modeled signal events from other workflow instances, system events like *task X started* or they may come from outside the workflow engine, for instance from a CEP engine. As an example, our maintenance workflow should immediately react on a detected *customer solvency warning* event, but only if the customer is not ranked as an A-customer for the maintenance service provider. If so, even already added or replaced spare parts should be revoked to reduce the risk of customer illiquidity. That means, at this point we want to combine workflow context-awareness and event-driven reactivity. The difference to specifying classic event-condition-action (ECA) rules is that we propose to use higher-level event-driven adaptation patterns for such cases, that directly rely on BPMN2 semantics. A corresponding draft for such a pattern catalogue is shown in table 1. Please note, that we do not aim at re-inventing or standardizing workflow or rule semantic. Such a pattern catalogue is rather intended to be a reference base for looking up solutions to modeling problems of event-based adaptations within workflow, similar to what are design patterns for software engineering or the workflow patterns [VTKB03] for workflow modeling itself.

Two pattern examples and their application to parts of the example workflow are shown in figure 2, while figure 3 contains the whole adapted worflow. Pattern #2 in combination with rule #6 realizes the already mentioned revoking of spare parts if a non-top customer is threatened to become illiquid. Pattern #1 in combination with rule #5 guarantees that for a particular engine designer, all measurement activities take place within a particular time frame. Depending on the dockyard location, there may be different policies for how long the ship may run its engine while it is docked. If this time frame is exceeded, the measure devices need to be reset and the measurements have to be canceled and repeated.

```
RULE #5: IF dockyardStation==Hamburg ADD nofailabletimedhandler AROUND measurements WITH 2h HANDLER resetDevices
RULE #6: IF customerRelevancyRanking>A
```

```
ADD escalatedCompensation ON customerSolvencyWarning AROUND sparePartsReplacement
```

With the presented approach, we have extended the concept of rule-based variant creation with context-dependent event-based reaction semantics. We thereby keep the LHS of rules as well as the language constructs of their RHS on a minimum of complexity, relying on well defined modeling change operations as well as reusable and parameterizable higher level event-based adaptation patterns.

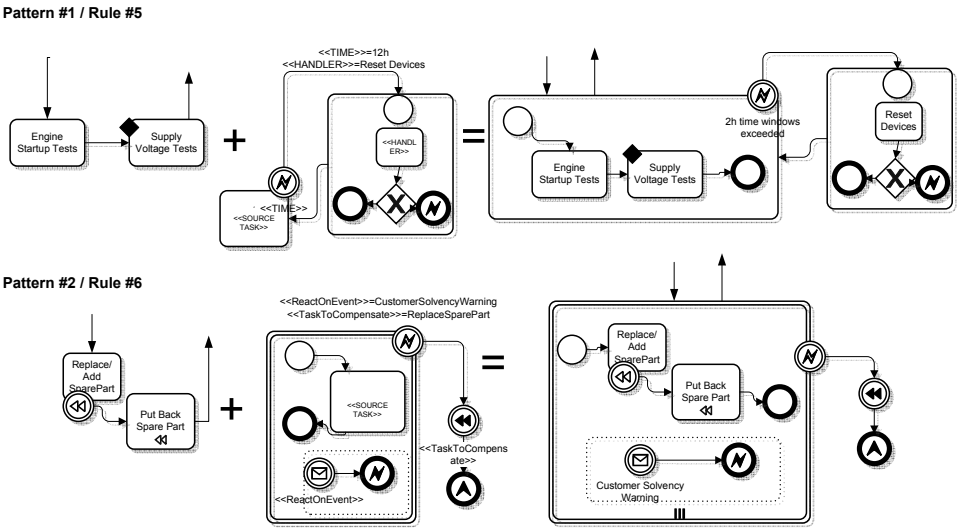

Figure 2: Example Event-Driven Adaptation Patterns and Application to Workflow Regions

5 Implications and Challenges

Although the presented rule-based methods for modeling context-aware and reactive workflows provide a high level of flexibility, they also entail a number of general implications and challenges one needs to be aware of. The most relevant of them are briefly discussed in the following:

- The determination and modeling of the *context* itself is a research challenge for its own. That means, it is required to develop a tough understanding of what are the context factors that cause a process to change and which of them are suitable for being considered in a process-aware enterprise information system, as e.g. discussed in [PPS+09].
- The increased degrees of freedom regarding modeling decisions have to be accompanied by appropriate modeling guidelines. For example, one question to decide is

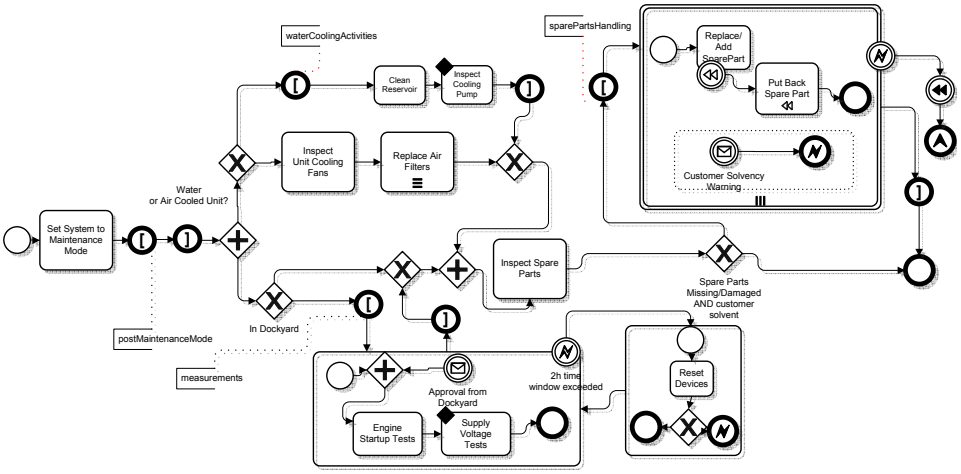

Figure 3: Adapted Maintenance Workflow

which granularity should be chosen to model a *basic workflow*, e.g. as an intersection, a cross-section or a union of all possible instance configurations [HBR09b]. Related to this is the question of where to best model decision logic, within the workflow or within rules [zMIK08]. [DKGZ10] point out the challenge that e.g. a context variable change could be represented and captured as an event and vice versa, who also has to be addressed by modeling guidance and a supporting environment.

- The assurance of design-time and runtime consistency of the workflow becomes significantly harder, since contradicting change operations need to be eliminated. [HBR09b] contains an overview of relevant aspects for correctly dynamically configuring variants and concrete suggestions for addressing respective issues.

- One definite strength of static workflow models which do not change at runtime is their relatively easy monitoring, traceability and analysis. Providing such features for rule-based adaptive processes that furthermore build on eventing semantics requires far more sophisticated approaches. Potentials lie in the analysis and mining of workflow execution logs, as e.g. implemented in ProM [VVG+09].

- A convenient modularization and structuring of the adaptation pattern catalogue itself is required. For example as one can see in table 1, pattern #1 is in fact a specialization of pattern #4. It would be useful to explicitly define such associations to allow the modeler a comfortable exploration of the catalogue.

6 Prototypical Realization in jBoss Drools

A part of the presented example workflow and the concepts presented in this work has been prototypically implemented in jBoss Drools[4] [Bal09], an integrated platform for business logic modeling and execution. Drools consists of a workflow component as well as a RETE-based rule component together with a temporal algebra extension for CEP.

The flow component is directly capable of importing and executing a subset of XML-serialized BPMN2. As however the required variant points for the adaptation mechanisms specified in section 3 and 4 are not defined by BPMN2, we convert them to intermediate throw event message nodes and catch the events within the rule engine for processing. For a variant point attached to a single task, we create one message node before and one message node after the corresponding task. The messages are used to signal the Drools runtime environment when a variable part of the workflow is either entered or left. In figure 4 in the left part, one can see a part of our example workflow with the corresponding variant points converted to message nodes and the applied pattern #1. The context-aware adaptation rules as presented before are converted into an ECA format as shown in the right part of the figure. That means, if a variable part of a workflow is entered, only those rules are triggered that have a potential impact on the respective part of the workflow by catching the entering event of the variation point.

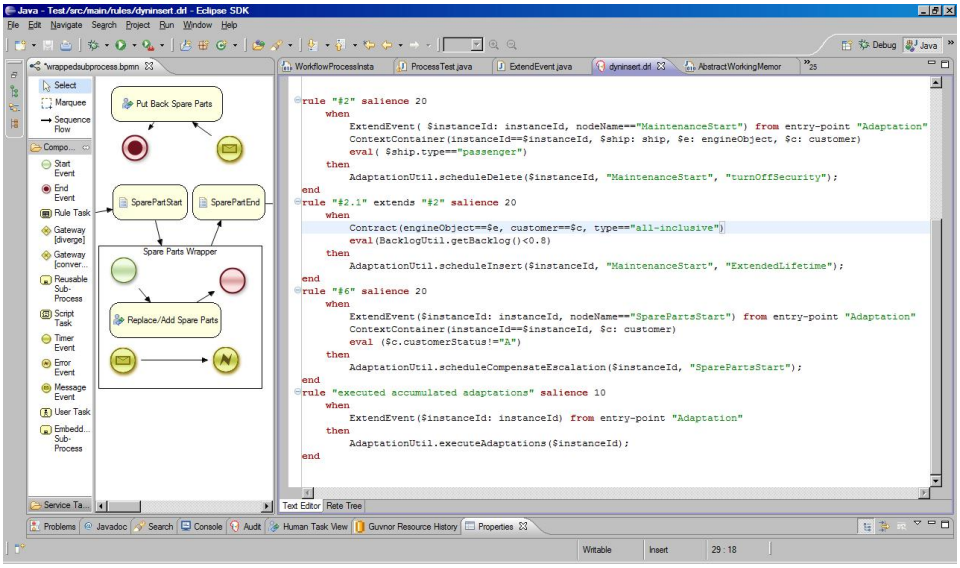

Figure 4: Screenshot of Example Workflow and Adaptation Rules in jBoss Drools

The remaining LHS of the rules query the (global or instance-specific) context variables and the RHS add the corresponding change operations into a change set. An applica-

[4] http://www.jboss.org/drools

tion rule with a lower salience (causing a firing only after the adaptation rules have fired) carries out the change operations in the set. Since Drools does not maintain any linkage (flow) information between instantiated activities in the runtime, the actual workflow instance adaptation is currently achieved by creating a model copy of the workflow segment enclosed by the variant point markers (constrained to be block-structured as mentioned before). To this model copy, the adaptations are applied. The one single node at the end of the segment is then connected to the ending variant marker via a converging XOR gateway. If a flow from an adapted segment reaches the ending variant point, the nodes that have been copied only to realize the adapted segment can be traced back and removed from the model via a *cleanUp()* method reacting on the event thrown by the ending variant node. The procedure for adding and removing adapted segments is visualized in Figure 5.

As a lessons learned, we can state that the current realization in jBoss Drools bears some intricateness which mainly results from the absence of workflow graph information on the workflow instance level. For this reason, we will also consider other implementation variants in the future, for example by more elegantly translating the variable parts of a workflow which can be subject to adaptation to a set of rules which can be handled and changed easier at runtime.

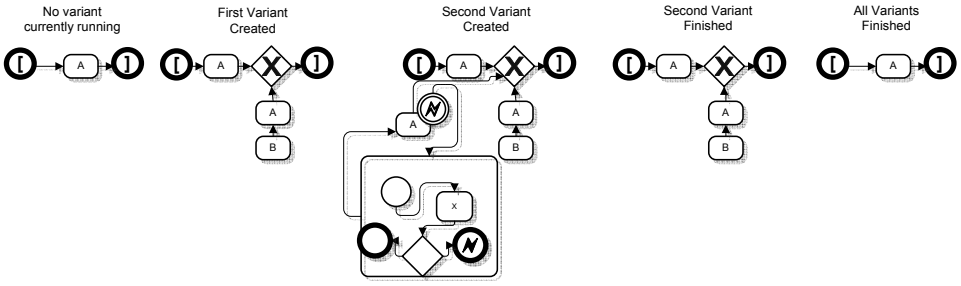

Figure 5: Adapted Segment Copies Added/Removed to the Model when Tokens Arrive at Variant Start/End Nodes

7 Conclusion

This work was motivated by the requirement for modern process-aware enterprise information systems to support context-dependent, highly variant and dynamically changing workflows. We have introduced an example service delivery workflow for ship engine maintenance and demonstrated how context-rules can be used to apply change operations and dynamically tailor the workflow to changing contexts. It has however also been shown that such simple rule-based adaptations are sometimes not sufficient when eventing paradigms are largely involved in a modeled workflow and context-dependent reactivity and exception handling has to be realized. For this purpose, we have proposed the elaboration of workflow change operations with an event-driven adaptation pattern catalogue. One such pattern for example is the context-dependent cancellation of a workflow seg-

ment if a specific event occurs and the subsequent triggering of a particular handler task. The practical feasibility and testability of our approach has been substantiated by partly extending an open-source rule- and workflow-engine with the presented functionality.
For future work, the proposed pattern catalogue has to be extended by analyzing typical use-cases also other than maintenance where eventing plays a large role within a workflow, for example logistics. Furthermore, other implementation variants than the direct manipulation of flow graphs, for example by mapping some workflow constructs to ECA rules, have to be considered. Finally, an intuitive modeling environment that allows for a comfortable browsing of patterns and their integration in rules is required. For such a modeling environment, existing concepts to provide design-time validation and verification of a model have to be examined regarding their applicability to our approach.

Acknowledgements

The work presented in this paper was developed in the context of the project *Allianz Digitaler Warenfluss* (ADiWa) that is funded by the German Federal Ministry of Education and Research. Support code: 01IA08006.

References

[AHK05] Patrick Albert, Laurent Henocque, and Mathias Kleiner. Configuration based workflow composition. In *ICWS '05*, pages 285—-292. IEEE, 2005.

[ATEV06] Michael Adams, Arthur H M Ter Hofstede, David Edmond, and Wil M P Van Der Aalst. Worklets : A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In *CoopIS '06*, pages 291–308. Springer, 2006.

[Bal09] Michal Bali. *Drools JBoss Rules 5.0 Developer's Guide*. Packt Publishing, 2009.

[BGP07] L. Baresi, S. Guinea, and L. Pasquale. Self-healing BPEL processes with Dynamo and the JBoss rule engine. In *International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting*, number 2. ACM, 2007.

[CCF+00] F. Casati, S. Castano, M. Fugini, I. Mirbel, and B. Pernici. Using patterns to design rules in workflows. *IEEE Transactions on Software Engineering*, 26(8):760—-785, 2000.

[DKGZ10] M. Döhring, L. Karg, E. Godehardt, and B. Zimmermann. The Convergence of Workflows, Business Rules and Complex Events. In *ICEIS '10*, Funchal, Portugal, 2010.

[GvJV07] F. Gottschalk, W.M.P. van Der Aalst, and M.H. Jansen-Vullers. SAP WebFlow made configurable: Unifying workflow templates into a configurable model. In *BPM*, volume 4714. Springer, 2007.

[HBR09a] A. Hallerbach, T. Bauer, and M. Reichert. Configuration and management of process variants. *Handbook on Business Process Management, Springer-Verlag*, 2009.

[HBR09b] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Correct Configuration of Process Variants in Provop, UIB-2009-03, 2009.

[HBR10] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Capturing variability in business process models: the Provop approach. *Software Process: Improvement and Practice (accepted for publication)*, 2010.

[KRSRS96] G Kappel, S Rausch-Schott, W Retschitzegger, and M Sakkinen. From Rules ro Rule Patterns. In *CAISE '96*, pages 99—-115. Springer, 1996.

[KY09] Akhil Kumar and Wen Yao. Process Materialization Using Templates and Rules to Design Flexible Process Models. In *RuleML '09*, pages 122–136, Las Vegas, Nevada, 2009. Springer.

[LDKD09] M. La Rosa, M. Dumas, R. Kaarik, and R. Dijkman. Merging Business Process Models. *QUT Prints*, 2009.

[LSGY09] Ruopeng Lu, Shazia Sadiq, Guido Governatori, and Xiaoping Yang. Defining Adaptation Constraints for Business Process Variants. In *BIS '09*, page 145â156. Springer, 2009.

[Lun06] A. Lundberg. Leverage Complex Event Processing to Improve Operational Performance. *Business Intelligence Journal*, 11(1):55, 2006.

[LVDT08] M. La Rosa, W.M.P. Van Der Aalst, Marlon Dumas, and A.H.M. Ter Hofstede. Questionnaire-based variability modeling for system configuration. *Software and Systems Modeling*, 8(2):251—-274, 2008.

[PPS+09] K Ploesser, M Peleg, P Soffer, M Rosemann, and J.C. Recker. Learning from Context to Improve Business Processes. *BPTrends*, 6(1):1—-7, 2009.

[PSWW05] F. Puhlmann, A. Schnieders, Jens Weiland, and M. Weske. *Variability mechanisms for process models*, volume 17. 2005.

[RV07] M Rosemann and WMP Van Der Aalst. A configurable reference modelling language. *Information Systems*, 32(1):1—-23, 2007.

[SKY06] Shuang Sun, Akhil Kumar, and John Yen. Merging workflows: A new perspective on connecting business processes. *Decision Support Systems*, 42(2):844—-858, 2006.

[vdADG+08] W. van der Aalst, Marlon Dumas, Florian Gottschalk, A. ter Hofstede, M. La Rosa, and Jan Mendling. Correctness-preserving configuration of business process models. *Fundamental Approaches to Software Engineering*, 4961:46—-61, 2008.

[vPS09] W.M.P. van Der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows - Balancing between flexibility and support. *Computer Science - Research and Development*, 21(2):99–113, 2009.

[VTKB03] Wil Van Der Aalst, Arthur H.M. Ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5—-51, 2003.

[VVG+09] W.M.P. Van Der Aalst, B.F. Van Dongen, C. Günther, A. Rozinat, H. M. W. Verbeek, and A. J. M. M. Weijters. ProM : The Process Mining Toolkit. In *BPM '09 (Demonstration Track)*. CEUR-WS.org, 2009.

[zMIK08] M. zur Muehlen, Marta Indulska, and Kai Kittel. Towards Integrated Modeling of Business Processes and Business Rules. In *19th Australasian Conference on Information Systems*, 2008.