The DPIL Framework: Tool Support for Agile and Resource-Aware Business Processes*

Stefan Schönig and Michael Zeising

University of Bayreuth,
Universitätsstraße 30, 95447 Bayreuth, Germany
{stefan.schoenig,michael.zeising}@uni-bayreuth.de

Abstract. The Declarative Process Intermediate Language (DPIL) is a declarative process modelling language that allows for specifying multi-perspective and multi-modal agile, i.e., flexible, processes. The expressiveness of DPIL and its suitability for business process modelling have been evaluated with respect to the well-known Workflow Patterns. The DPIL Framework provides a tool set for supporting agile and resource-aware business processes based on the language DPIL. While the DPIL Modeller component is used to create and verify models, the DPIL Navigator depicts a rule-based execution engine for enacting models. It comprises a web-based worklist which allows process participants to choose and perform tasks. In addition to this, the DPIL Miner component allows for the discovery of DPIL models from event logs.

1 Background and Significance to BPM

Two different representational paradigms for business processes can be distinguished: procedural models describe which activities can be executed next and declarative, i.e., rule-based models define execution constraints that the process has to satisfy [1]. In flexible processes the exact flow of activities cannot be fully determined at design time. These processes heavily depend on human participants, their decisions and expert knowledge. These information cannot be identified and formalized in a whole. As a result, these processes require highly flexible IT support. Flexible processes are common in healthcare where, e.g., patient diagnosis and treatment processes require flexibility to cope with unanticipated circumstances. In brief, a more flexible business process and IT support means a greater number of alternative paths. Within a rule-based model initially all paths are considered viable. The more constraints are added to the model the less paths remain. As result, to make a rule-based model more flexible constraints have to be removed or weakened. Moreover, a rule-based model focuses on crosscutting relations instead of the flow of activities. Hence, a rule-based approach is well-suited for modelling flexible processes [1, 2]. As mentioned above, declarative modelling provides means for increasing the number of alternative

* Copyright ©2015 for this paper by its authors. Copying permitted for private and academic purposes.

```
use group Student

process BusinessTrip {
   task Apply for trip
   task Approve Application
   task Book flight

   ensure sequence(Apply for trip, Approve Application)
   advice "Approval before booking recommended":
          roleSequence(Approve Application, Book flight, Student)
   advice "Flight should be booked by applicant":
          binding(Apply for trip, Book flight)

   milestone "Done": complete(of ApproveApplication) and
                     complete(of BookFlight)
}
```

Fig. 1: Process for trip management modelled with DPIL

paths without explicitly modelling them. Given the vast amount of alternatives, a differentiation between mandatory and only recommended actions in the form of modalities is reasonable [3]. Independent from a specific modelling paradigm different perspectives on a process exist. The organizational perspective, often also referred to as resource perspective deals with the definition and the allocation of human and non-human resources to activities. Due to the importance of human decision-making and expert knowledge, especially flexible processes need to explicitly integrate the organisational perspective [2]. In many declarative languages, however, an adequate representation of organisational patterns is often still not possible. The *Declarative Process Intermediate Language (DPIL)* [4] is a declarative process modelling language that supports the requirements described above. Unlike other declarative languages it is multi-perspective, i.e., it allows for representing several business process perspectives, namely, control flow, data and especially resources. In order to express complex organisational relations, DPIL builds upon a generic organisational meta model. The expressiveness of DPIL and its suitability for business process modelling have been evaluated [4] with respect to the well-known Workflow Patterns. DPIL supports 52% of the behavioral, 62% of the organizational and 75% of the informational patterns. Like this, DPIL meets around 50% more of the requirements than BPMN does. Moreover, it is multi-modal, meaning that it allows defining two different types of rules: rules representing mandatory relations (called *ensure* in DPIL) and rules representing recommended relations (called *advice* in DPIL). The latter are useful, e.g., to reflect good practices. Here, the modeller can give further explanations that are offered to the user during execution.

DPIL provides a textual notation based on the use of *macros* to define reusable rules. For instance, the *sequence* macro ($sequence(a, b)$) states that the existence of a *start event* of task b implies the previous occurrence of a

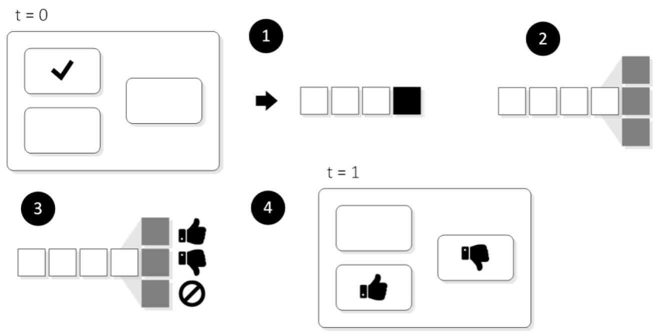


Fig. 2: Principle of rule-based process execution of the DPIL Navigator

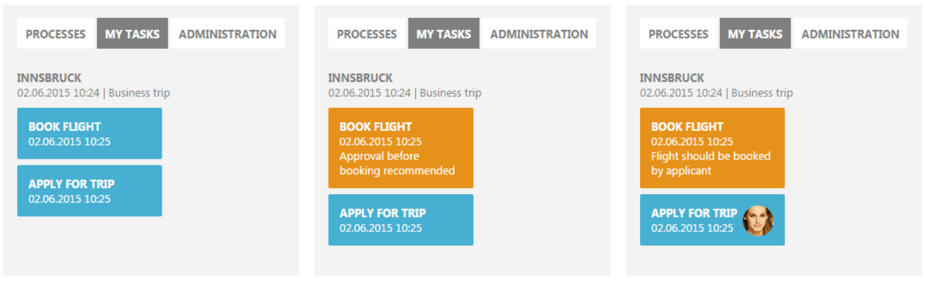
complete event of task a; and the *binding* macro ($binding(a, b)$) states that an activity b is assigned to the same actor that already performed activity a. The *roleSequence* ($roleSequence(a, b, r)$) finally depicts that the start of task b by an actor in role r requires the completion of task a before. Fig. 1 shows a model of a simple process for trip management in DPIL. It states that it is mandatory for all applicants to apply for a business trip before it can be approved. Moreover, it is recommended for students that the approval is carried out before a flight is booked. Since the applicant usually knows appointments best it is recommended that she books the flight herself. The DPIL Framework provides a tool set for modelling, executing and analysing flexible and resource-aware business processes based on the language DPIL as described above.

2 DPIL Framework: Overview and Demo Guidelines

By means of the example process above, the demo will show the functionality of the three modules the framework consists of: the *DPIL Modeller*, the *DPIL Navigator* and the *DPIL Miner*. The framework supports defining models using a textual editor based on the Eclipse IDE, the DPIL Modeller. The modeling tool support users through syntactic, semantic and qualitative model analysis based on the Xtext framework[1]. After specifying DPIL models like in Fig. 1 the model is compiled to an executable file that can be enacted by a rule-based execution engine, the DPIL Navigator.

The DPIL Navigator is based on the execution principle as visualized in Fig. 2. Tasks of a DPIL process model undergo a life cycle composed of events that is managed by the engine. A task, e.g., can be started and completed. The current state of a process is then the series of past events (1). Besides the static elements like human tasks, e.g., *Apply for trip* and *Approve application*, a process model may specify rules constraining that series of events, e.g., *sequence(Apply for trip, Approve application)*. When the model is executed, the engine simulates one event ahead for every model element (2) and evaluates the resulting series of

[1] http://www.eclipse.org/Xtext/



INNSBRUCK
02.06.2015 10:24 | Business trip

BOOK FLIGHT
02.06.2015 10:25

APPLY FOR TRIP
02.06.2015 10:25

INNSBRUCK
02.06.2015 10:24 | Business trip

BOOK FLIGHT
02.06.2015 10:25
Approval before
booking recommended

APPLY FOR TRIP
02.06.2015 10:25

INNSBRUCK
02.06.2015 10:24 | Business trip

BOOK FLIGHT
02.06.2015 10:25
Flight should be booked
by applicant

APPLY FOR TRIP
02.06.2015 10:25

(a) Sequence rule (b) RoleSequence rule (c) Binding rule

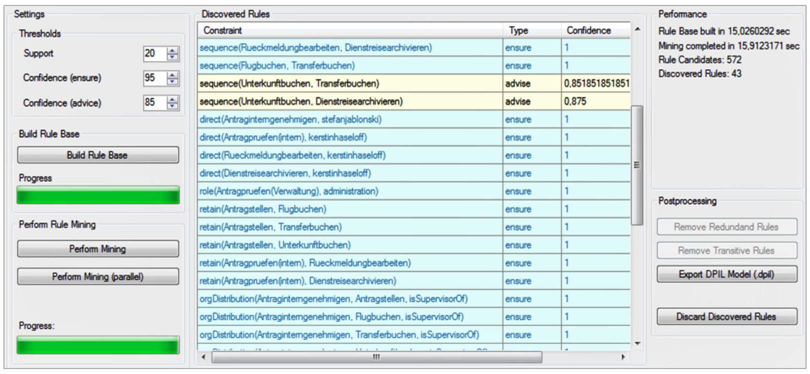
Fig. 3: Worklist of the DPIL Navigator in different situations

events on the basis of the rules (3). Each simulated event that does not violate any *ensure* rule is related to an action that the engine interprets immediately (4). A simulated start of a task by a certain participant, e.g., is interpreted as the assignment of this task to the participant. If the start event violates an *advice* rule, the action is marked as not recommended. The principle is demonstrated w.r.t. two different instances "Business trip to Innsbruck" of the example process. Fig. 3a shows the web-based worklist for a non-student, e.g., a professor. The task *Approve Application* is not visible since its execution violates the mandatory (*ensure*) *sequence* rule. The execution of the other tasks, however, is possible. Fig. 3b depicts the worklist for a student. In case of a user in role *Student*, the execution of the task *Book flight* violates the recommended (*advice*) *roleSequence* rule. Note, that the execution of the task is still possible, however, the violation of the rule is highlighted through orange colour and the indication text from the model. Fig. 3c shows the task worklist for a third person in the situation where the trip application has already been started by the professor. In this case the execution of *Book flight* violates the recommended *binding* rule, i.e., the flight should be booked by the applicant herself.

As well as manually creating a DPIL model using the DPIL modeler, users can also discover models from event logs using the DPIL Miner [5]. Here, the analyst selects the DPIL macros that should be discovered w.r.t. the provided event log. Fig. 4 shows the user-interface of the DPIL Miner with some discovered organizational patterns. Both the execution engine and the mining module are based on the JBoss Drools platform [6] that provides a current implementation of the *rete* rules solver [7]. Therefore, in both components DPIL rules are transformed to the Drools Rule Language (DRL).

3 Maturity and Future Work

The DPIL Framework is a well-evaluated prototype that is already used for demonstration purposes in academia as well as in industry. It has been developed and used in the KpPQ project. In [5] we show how it is possible to discover DPIL models from real-life event logs. The DPIL Navigator with the example process



Support: 20
Confidence (ensure): 95
Confidence (advice): 85

Constraint	Type	Confidence
sequence(Rueckmeldungbearbeiten, Dienstreisearchivieren)	ensure	1
sequence(Flugbuchen, Transferbuchen)	ensure	1
sequence(Unterkunftbuchen, Transferbuchen)	advise	0,851851851851
sequence(Unterkunftbuchen, Dienstreisearchivieren)	advise	0,875
direct(Antraginterngenehmigen, stefanjablonski)	ensure	1
direct(Antragpruefen(intern), kerstinhaseloff)	ensure	1
direct(Rueckmeldungbearbeiten, kerstinhaseloff)	ensure	1
direct(Dienstreisearchivieren, kerstinhaseloff)	ensure	1
role(Antragpruefen(Verwaltung), administration)	ensure	1
retain(Antragstellen, Flugbuchen)	ensure	1
retain(Antragstellen, Transferbuchen)	ensure	1
retain(Antragstellen, Unterkunftbuchen)	ensure	1
retain(Antragpruefen(intern), Rueckmeldungbearbeiten)	ensure	1
retain(Antragpruefen(intern), Dienstreisearchivieren)	ensure	1
orgDistribution(Antraginterngenehmigen, Antragstellen, isSupervisorOf)	ensure	1
orgDistribution(Antraginterngenehmigen, Flugbuchen, isSupervisorOf)	ensure	1
orgDistribution(Antraginterngenehmigen, Transferbuchen, isSupervisorOf)	ensure	1

Rule Base built in 15,0260292 sec
Mining completed in 15,9123171 sec
Rule Candidates: 572
Discovered Rules: 43

Fig. 4: User-interface of the DPIL Miner during the analysis of an event log

deployed is accessible at `http://dpilnavigator.kppq.de`. Furthermore, different screencasts illustrating the different features of the DPIL Framework are accessible online: the Modeller at `http://modeller.kppq.de`, the navigator at `http://navigator.kppq.de` and the miner at `http://miner.kppq.de`. Future work will focus on the implementation of DPIL model verification techniques and model simulation features that should support users when choosing from a diversity of possible action through look ahead strategies and recommendations.

References

1. D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal, “Declarative versus imperative process modeling languages: The issue of understandability,” in *Enterprise, Business-Process and Information Systems Modeling*, pp. 353–366, 2009.
2. R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya, “Declarative business artifact centric modeling of decision and knowledge intensive business processes,” in *Enterprise Distributed Object Computing Conference (EDOC)*, no. Edoc, pp. 151–160, 2011.
3. G. Regev and A. Wegmann, “A regulation-based view on business process and supporting system flexibility,” in *Advanced Information Systems Engineering*, vol. 5, pp. 91–98, 2005.
4. M. Zeising, S. Schönig, and S. Jablonski, “Towards a Common Platform for the Support of Routine and Agile Business Processes,” in *Collaborative Computing: Networking, Applications and Worksharing*, 2014.
5. S. Schönig, C. Cabanillas, S. Jablonski, and J. Mendling, “Mining the Organisational Perspective in Agile Business Processes,” in *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2015.
6. T. JBoss Drools, “JBoss Drools Documentation - Chapter 7: Rule Language Reference,” 2013.
7. C. Forgy, “Rete: A fast algorithm for the many pattern/many object pattern match problem,” *Artificial intelligence*, vol. 19, no. 1, pp. 17–37, 1982.