MuDePS: Multi-perspective Declarative Process Simulation

Lars Ackermann and Stefan Schönig

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

Abstract. Business process simulation supports the improvement and analysis of business process models. Especially log generation features gain more and more attractivity, for instance, because of their applications in the evaluation of process mining techniques. Additionally, cognitive science has shown that examples promote the comprehension of abstract models. This is important especially for declarative modeling languages, which are able to cope with highly flexible processes while mostly being less intelligible than imperative counterparts. Since most of the available process simulation tools are tailored to imperative languages they are not suitable in declarative contexts. The tool presented in this paper is able to simulate declarative, multi-perspective process models. It builds on DPIL, a declarative process modeling language. DPIL models can be created and compiled to a simulation model using the DPIL Modeler. MuDePS, a second component, is able to generate event logs that conform to the original DPIL model. Each generated log is able to describe an exhaustive, distinct set of valid process execution traces of a desired length.

Keywords: process simulation, log generation, declarative modeling

1 Background and Significance to BPM

Business processes differ regarding their strictness and level of governance. Thus, two paradigms for business process modeling emerged: Imperative models describe allowed process instances using step-by-step flow descriptions, whereas declarative, i.e. rule-based models consist of constraints that each instance has to satisfy [1]. Declarative process modeling languages (DPMLs) are based on the foundational principle that all execution paths are allowed as long as they are not explicitly forbidden. Rules are used to form a forbidden region for process executions. Hence, more flexible processes require less rules, i.e., the resulting model is comparably small and, thus, rule-based approaches are well-suited for the representation of flexible processes [1, 2].

```
use group Administration

process BusinessTrip {
    task Apply for Trip
    task Approve application
    task Book accommodation

    document Application

    ensure produces(Apply for Trip, Application)
    ensure consumes(Approve application, Application)
    ensure sequence(Approve application, Book accommodation)
    ensure role(Approve Application, Administration)
    ensure binding(Book accommodation, Apply for Trip)

    milestone "Done": event(of Book accommodation)
}
```

Fig. 1: Process for trip management modeled with DPIL

Orthogonal to the two modeling paradigms a process involves multiple *perspectives*. Activities in the process, for instance, follow a certain order (*functional* and *behavioral*), are performed by certain organizational resources (*organizational*) and require an appropriate handling of data (*data-oriented*) [2, 3]. Many DPMLs are limited to the functional and the behavioral perspectives. Even the more expressive *Dynamic Condition Response Graph* [4] language still lacks the involvement of organizational resources. In contrast, the *Declarative Process Intermediate Language (DPIL)* [3] has been designed for modeling process rules for *multiple* perspectives. Thus, using DPIL it is possible to model dependencies between activities as well as between resources and activities and all other combinations of elements from different or the same perspectives. DPIL and its expressiveness have been motivated and evaluated in both, industry applications like the KpPQ project[2] and in academical contexts [3] based on the well-known set of *Workflow Patterns*. One result of the academical investigation was that DPIL provides support for around 50% more of the requirements than BPMN does. However, though we decided to use DPIL as example language the principle can be applied to other DPMLs, too [5].

DPIL is a textual modeling language and allows for defining reusable rule templates (*macros*) in order to keep the resulting model as concise as possible. For instance, the *produces* macro (*produces(t,d)*) states that an activity t cannot be complete until a data object d has been produced. Consequently, the *consumes* macro (*consumes(u,d)*) stipulates that a second activity, u, cannot be started until this data object d is available. This results in a temporal ordering of the two activities t and u driven by a data dependency. Sometimes the temporal ordering of activities might depend on information which are not part of the process. Such constraints can be formulated using the *sequence* macro (*sequence(t,u)*), whereby u must be preceded by at least one execution of t. Resources can be assigned to an activity using the *role* macro (*role(t,r)*), which restricts an actor of t to be in role r. Such assignments do not need to be static, as the *binding* macro (*binding(t,u)*) shows. This macro says that the actor in the

[2] For more information we would like to refer to: `http://kppq.de/`

activities t and u has to be the same. The DPIL model in Fig. 1 is a simplified sketch of a business trip application process. It says that an application can be approved by the administration as soon as the application document is available. The accommodation has to be booked by the applicant and, at the earliest, as soon as the application has been approved. This step concludes the process.

Process model analysis and improvement phases can be supported by business process simulation techniques [6]. Simulation tools can produce log files which can be further analyzed enabling the user to predict the performance of a process in an operative context. As a second application, simulated logs can be used to gain a deeper understanding of the meaning and properties of a process. Cognitive research has shown that examples help to improve the comprehension of rules and abstractions [7]. A third advantage of simulation tools that produce logs is the opportunity to evaluate process mining techniques, as it has been shown in [8]. However, though there are many simulation techniques for *imperative* languages there is currently a lack of simulation tools for *declarative* languages. The approach in [8] is able to simulate declarative process models but the underlying DPML is limited to the functional and behavioral perspectives of processes. The transformation from multi-perspective declarative process models to an imperative representation is still vague. Consequently, it is vague, too, whether imperative simulation tools can be applied. We therefore complement the DPIL framework [9] with a multi-perspective declarative process simulation tool called *MuDePS* which is presented in the remaining part of the paper.

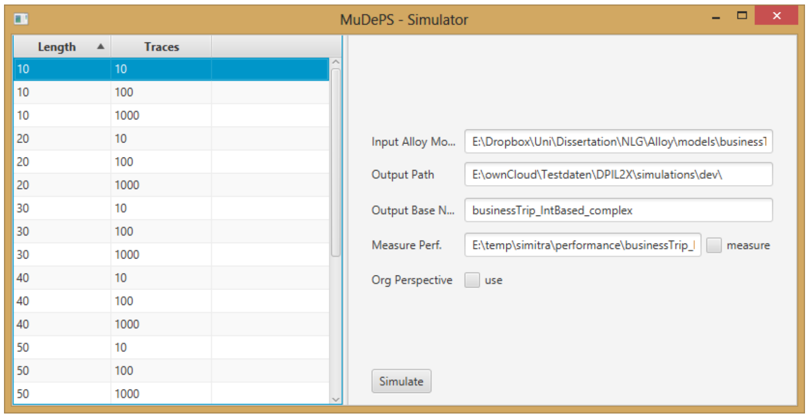
2 MuDePS: Overview and Demo Guidelines

In the demo we make use of two components of the DPIL framework, namely *DPIL Modeler* and *MuDePS* itself in order to demonstrate the simulation of the example process above as well as an extended version. The modeler ships with a textual editor based on the *Eclipse IDE* in combination with *Xtext*. Thus, the user is supported through general IDE features like auto-completion as well as through language features like syntax checking and reference resolution. MuDePS operates on models formulated using a logic language called *Alloy* [10]. The gap between a model specified in DPIL and its Alloy representation is closed using an automated model-to-text generator based on *Acceleo*.[3]

The MuDePS simulation tool works as follows. First, a DPIL model is translated into an Alloy-conform specification. Alloy ships with an analyzer which is able to identify *unique* examples for a given model and target size *exhaustively*. This means that each possible process execution path occurs exactly once in the resulting log which is serialized according to the *eXtensible Event Stream (XES)* standard format[4]. This standard is based on the idea of *Discrete-Event Simulation* which means that the log of a particular process execution is a chain of events whereby each encapsulates all relevant information about the executed activity and the executing resource and tools.

[3] Xtext: `www.eclipse.org/Xtext/`, Acceleo: `www.eclipse.org/acceleo`

[4] `http://www.xes-standard.org/`



Length	Traces
10	10
10	100
10	1000
20	10
20	100
20	1000
30	10
30	100
30	1000
40	10
40	100
40	1000
50	10
50	100
50	1000

Input Alloy Mo...	E:\Dropbox\Uni\Dissertation\NLG\Alloy\models\business1
Output Path	E:\ownCloud\Testdaten\DPIL2X\simulations\dev\
Output Base N...	businessTrip_IntBased_complex
Measure Perf.	E:\temp\simitra\performance\businessTrip_I ☐ measure
Org Perspective	☐ use

Fig. 2: MuDePS: User-interface of the simulation tool

A simulation model in MuDePS consists of two main parts: *(i)* A static part that describes the general structure and constraints for an intrinsically valid process event trace and *(ii)* a dynamic part consisting of constraints that have been generated from the source DPIL model. The static part ensures that each event occurs at a particular point in time (behavioral perspective) and that each event is able to encapsulate the activity name (functional perspective), the performer (organizational perspective) and data access information. An actor is usually part of an organizational structure. Thus, we also provide an Alloy implementation of the well known organizational meta-model discussed in [11].

The dynamic part contains the Alloy representation of the particular process model like the example in Fig. 1. The group, task and document elements are transformed to so called *signatures* which are comparable to *classes* in object-oriented programming languages. Thus, signatures can be abstract, too, and a signature that represents a process model element is an extension of one signature from the static part. A process rule is transformed to a so called *fact*. A fact is an invariant, i.e. a constraint that always must be fulfilled and that can be used to represent non-structural restrictions. In order to run the simulation we make use of Alloy's *run* command for which we have to provide a maximum target size of the solutions, i.e. the maximum process trace length. Running the command, the Alloy analyzer translates the Alloy model into a system of equations and applies a solver. MuDePS iterates over the resulting solutions considering the provided maximum trace length which, in turn, makes the solution space finite.

Based on its characteristics MuDePS can be used to produce an *exhaustive* set or a set of a *desired size* of example traces for a given DPIL model and maximum trace length. The traces are provided in the XES log format and can be post-processed by any desired application that operates on XES files.

3 Maturity and Future Work

As already shown in [9], the DPIL Framework is a well-evaluated prototype that is used for demonstration purposes in academic and industrial contexts.

The new part of the framework, MuDePS, shown in Fig. 2, has been evaluated regarding correctness and performance in [12] based on a real-life DPIL model. MuDePS itself, installation instructions as well as an example process and an exemplary generated event log are available at `http://mps.kppq.de`. Additionally we provide a screencast showing the basic workflow for simulating a DPIL model with MuDePS at: `https://youtu.be/JhqSiAxChKQ`. Currently the transformation from DPIL to Alloy is done using the signature of the macros. However, this means a restriction regarding DPIL's flexibility to create new process rule templates. That is why we are currently working on a more flexible transformation based on the actual rule structure. Additionally, the computation time for larger process event logs (> 80 events) is rather low (around 45 minutes for 1000 instances). However, this is caused by the multi-perspectivity but can be optimized, e.g. through partially inverting constraints in order to early minimize the solution space. Future versions of the tool will also consider activity life cycles (event start/completion) and modalities, i.e. soft and hard constraints. Additionally it will be fully integrated into the DPIL Modeler.

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 *BPMDS*, 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 *EDOC*, pp. 151–160, 2011.
3. M. Zeising, S. Schönig, and S. Jablonski, "Towards a Common Platform for the Support of Routine and Agile Business Processes," in *CollaborateCom*, 2014.
4. T. T. Hildebrandt and R. R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," *PLACES 2010 (Journal v.)*, 2011.
5. L. Ackermann, S. Schönig, and S. Jablonski, "Simulation of multi-perspective declarative process models," in *Int. Conf. on BPM*, Springer, 2016.
6. W. M. P. van der Aalst, "Business Process Simulation Revisited," *Enterprise and Organizational Modeling and Simulation*, vol. 63, pp. 1–14, 2010.
7. A. L. Brown and M. J. Kane, "Preschool children can learn to transfer: Learning to learn and learning from example," *Cogn. Psychology*, vol. 20, pp. 493–523, 1988.
8. C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of declare models," in *EOMAS*, pp. 20–36, 2015.
9. S. Schönig and M. Zeising, "The DPIL framework: Tool support for agile and resource-aware business processes," in *BPM Demo Session*, pp. 125–129, 2015.
10. D. Jackson, *Software Abstractions: logic, language, and analysis.* MIT press, 2012.
11. C. Bussler, "Analysis of the organization modeling capability of workflow-management-systems," in *PRIISM*, pp. 438–455, 1996.
12. L. Ackermann, S. Schönig, and S. Jablonski, "Simulation of Multi-Perspective Declarative Process Models," *preprint: https://epub.uni-bayreuth.de/id/eprint/2881*, 2016.