Language-centric approaches for improving business process model acceptance

Lars Ackermann

University of Bayreuth, Germany
lars.ackermann@uni-bayreuth.de

Abstract Business process modeling is a crucial task in business process management. The plethora of process modeling languages available offers flexibility but often leads to language barriers, too. This pertains to, e.g., the comprehensibility of a model in case that a stakeholder is unfamiliar with the modeling language and to its compatibility with "foreign-language" process execution systems. Among others literature and several research projects suggest *(i)* a meticulously compiled set of process execution traces, *(ii)* a translation into a suitable language and *(iii)* a model description in natural language as means that cover the potential to solve these issues. This abstract and the associated dissertation describe approaches that automatically generate these artifacts.
Comparable trace generators solely operate on imperative or plain control-flow based process models. Translation techniques and natural-language-text generators for process models are rare and mostly limited to a specific, imperative language. The approaches are evaluated by their application to various exemplary models and through a qualitative survey.

Keywords: business processes model acceptance, trace generation, process model translation, natural language generation

1 Introduction

Modeling an organization's business processes has proved to be a crucial task in business process management (BPM). Process models often play a dual role, i.e. they are used as means of communication between human process stakeholders and as a specification for system-aided process enactment [1].

As a means of communication a model has to be comprehensible [2], requiring at least that the stakeholders are familiar with the modeling language used. Due to the plethora of highly diverse process modeling languages it is hard to get familiar with all of them. Especially the notions of imperative and declarative modeling often cause confusions [3]. Imperative languages specify all permitted process instances explicitly. In contrast declarative process modeling languages are used to describe process instances implicitly based on rules. However, comprehensibility for human stakeholders does not imply that a process model ships with a formal specification of its execution semantics (e.g. swimlanes in BPMN), that it is compatible with the desired business process execution system and not

even that it is a *formal* model at all [1]. Thus, if a process model is meant to play this dual role an adverse language selection or tradeoff between the two requirement dimensions might cause language barriers and, thus, acceptance issues.

The PhD thesis [4] associated with this abstract identifies three artifacts which were likewise considered to be useful in literature and in several research projects[1] in order to improve a process model's acceptance:

- A carefully selected set of intentionally valid/invalid *process execution traces*,
- a model *translation* into a desired target language and
- a *natural-language* process *description*.

The following discussions focus on approaches that derive the artifacts automatically from process models offering a scalable user-side control over the results.

Process execution traces are usually records of real-world process instances. A selected set of traces with certain properties (e.g. a particular activity has to be executed) in combination with the source model might help to understand the model's behavioral semantics. Additionally, several cognitive studies showed that learning formal specifications can be boosted by positive and negative examples. However, traces of real-world process instances have several drawbacks, including *uncontrolled* contents and noise and sometimes they are simply not available (e.g. due to privacy policies [5]). Generating them artificially (Sec. 2.1) shifts the control over contents and noise to the generation tool and, thus, to its user.

Translating a process model manually is a cumbersome and error-prone task [6]. Consequently translation systems for several process modeling languages have been developed. However, the overall coverage of a selection of 15 common process modeling languages is rather low (10% direct, 20% including transitive closure). Therefore, Sec. 2.2 describes a complementing translation method.

Natural language process descriptions can solve language barriers since, in contrast to formal languages, natural languages are a common means of communication in daily life. Compiling natural language process descriptions manually and keeping them consistent can be time-consuming and requires ongoing effort [7]. Natural-language-text generation (NLG) techniques can overcome this issue but there are none for declarative process models. Thus, the NLG approach for declarative process models sketched in Sec. 2.3 is a necessary complement.

2 Improving process model acceptance: Three approaches

This section summarizes the basic ideas and innovative facets of three approaches (Fig. 1) for generating the artifacts discussed in the introduction.

In a nutshell the main advantage of the trace generator is its ability to create traces with controllable contents for declarative process models that cover also process aspects beyond control flow (e.g. organizational or data-related dependencies). The translation approach for process models avoids a cumbersome, language-specific definition of mapping rules between source and target metamodel elements and is generic. This increases the overall coverage of supported

[1] e.g. the *C2P2* project (`http://kppq.de/`) that focused on tools for all BPM tasks

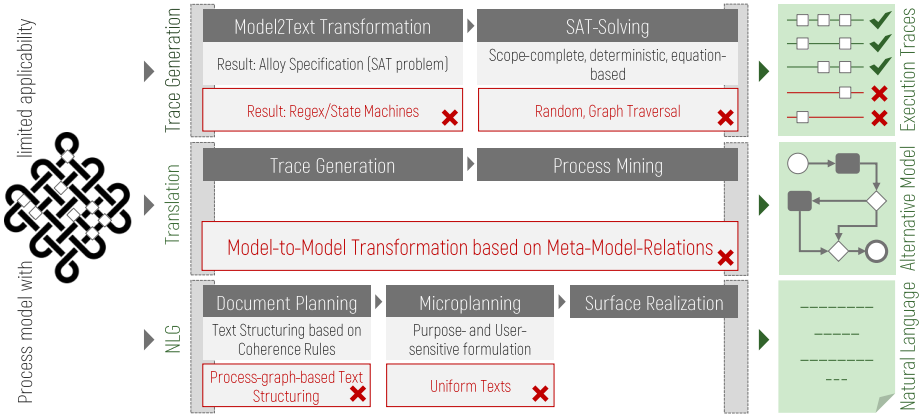

Fig. 1: Key differences compared to existing approaches

process modeling language pairs from 10 to 25%. Finally, the major innovation of the NLG approach is, that it operates on declarative process models.

2.1 Trace generation (MuDePS)

Trace generation means to simulate process executions in discrete time steps and to log each execution state as an event. An event is characterized by several attributes, e.g. a timestamp, an activity name and a snapshot of the current values of all process variables [8]. A collection of traces is called *event log*.

Most existing trace generators operate on imperative process models making them not applicable to declarative process models [8]. However, some approaches offer features that are also relevant for the approach discussed below *(MuDePS)*, e.g. *(i)* support for different process perspectives, *(ii)* configurable contents, intentional noise and *(iii)* a configurable initial state. Business process modeling languages differ widely in terms of their expressiveness which can be analyzed based on so called *process perspectives* [9]. *Single-perspective* languages, such as Petri nets, are limited to descriptions of the control flow. *Multi-perspective* notations (e.g. BPMN) also involve organizational and data-related information.

In [8] one of the few trace generators for declarative process models is described. It is based on regular expressions which is applicable to single-perspective, declarative models but cannot be applied to multi-perspective models. It would cause an alphabet of infinite size if a process model contains variables with continuous value ranges. A few other approaches operate on multi-perspective, declarative models but are unable to generate artificial noise because their simulation capabilities are limited to an exploration of the positive solution space.

MuDePS overcomes these drawbacks by translating the process model into the logic language *Alloy* [10] which means transforming the trace generation problem into a *satisfiability (SAT) problem*. An included analysis engine transforms the specification into a set of boolean equations that can be evaluated

by deterministic *SAT solvers*. By formulating the process rules as constraints the SAT solvers determine valid solutions for a configurable number of events per trace. Alternatively the rules can be formulated as assumptions causing the analysis engine to search for counter examples. Constraints can further be used to configure the initial state for the generation and desired trace contents.

2.2 Process model translation (SiMiTra)

The coverage of existing translation approaches for process models is rather low (Sec. 1). This issue is intensified considering that for n languages *(n^2-n)* transformation rule sets are needed to enable translations between all languages.

SiMiTra is a translation approach that discards the traditional principle that is based on meta-model transformation rules [6] and instead relies on combinations of existing trace generation and process mining techniques. It uses event logs as a transfer medium which decreases the overall number of required translation components to *2n*. Process mining approaches are algorithms that discover process models from event logs, usually based on co-occurrence thresholds for event attributes. These thresholds are used to compensate *undesired* noise in event logs of real-world process executions. However, logs in the SiMiTra translation system are noise-free as they are created by trace generators – assuming they work as expected. Consequently, all thresholds should be set as tolerantly as possible since most trace generators are based on local, random decisions which might cause infrequently occurring trace contents that would be filtered as noise otherwise. However, log completeness still cannot be guaranteed since process variables can have a continuous value range causing an infinite solution space. Additionally, the thesis [4] discusses limitations regarding the expressiveness of event logs, which are for instance, their inability to explicitly encode decision points or to differentiate between static and dynamic relations of process entities.

2.3 Natural language generation (NL4DP)

NL4DP a NLG approach for multi-perspective process models combines guaranteed model-text consistency with customization options based on a given user model and purpose. The backbone of the technique is a well-established pipeline model [7] consisting of three transformation phases. NLG techniques for imperative process models (e.g. [11]) make use of the native structuring properties of the directed-graph-based models themselves in order to compose a coherent text. However, this cannot be applied to declarative models and dedicated techniques for the latter are not available. Hence, the main contribution in the first phase *(Document Planning)* is an algorithm that groups related process rules based on rule patterns and common parameters and a transformation of the relevant information from the process model into formal sentence specifications. In the remaining two phases the results of the first phase are refined *(Microplanning)* by using different formulations depending on whether the generated texts are explanations of single rules during process execution or a complete description of the model for documentation purposes. In the former case the texts are also

customized in order to address a user explicitly if applicable. Finally, standard *Surface Realization* techniques derive one natural language sentences from each sentence specification which are then compiled to a text document according to the specified macrostructure.

3 Implementation and Evaluation

MuDePS and SiMiTra are implemented as an operator[2] and a set of process templates for the *RapidMiner* data mining platform. NL4DP is based on *Eclipse* and the *Acceleo* text generation plugin[3].

Since the trace generator relies on matured SAT solvers the translation of process rules into Alloy is the only error source which could be eliminated by means of Alloy-based test cases. SiMiTra has been evaluated based on conformance checking between the resulting model and an unseen log generated from the source model. The results are encouraging but differ depending on the complexity of the source model. NL4DP has been evaluated through its application to selected process models and comparing the generated with manually created texts. Finally, a qualitative survey presages that the three artifacts can be valuable means for understanding and analyzing declarative process models.

References

1. W. van der Aalst, *Process Mining: Data Science in Action.* Springer, 2016.
2. J. Mendling, H. Reijers, and W. van der Aalst, "Seven process modeling guidelines (7PMG)," *Information and Software Technology*, vol. 52, no. 2, pp. 127–136, 2010.
3. 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 *Ent., BP and Inf. Sys. Mod.*, pp. 353–366, Springer, 2009.
4. L. Ackermann, *Sprachzentrierte Ansätze zur Steigerung der Akzeptanz von Geschäftsprozessmodellen.* PhD thesis, University of Bayreuth, März 2018.
5. A. Burattin, "PLG2: Multiperspective Process Randomization with Online and Offline Simulations," in *Onl. Proc. of the BPM Demo Track*, CEUR-WS.org, 2016.
6. K. Czarnecki and S. Helsen, "Feature-based survey of model transformation approaches," *IBM Systems Journal*, vol. 45, no. 3, pp. 621–645, 2006.
7. E. Reiter, R. Dale, and Z. Feng, *Building natural language generation systems*, vol. 33. MIT Press, 2000.
8. C. D. Ciccio and M. L. Bernardi, "Generating Event Logs through the Simulation of Declare Models," in *EOMAS workshop*, pp. 20–36, Springer, 2015.
9. S. Jablonski and C. Bussler, *Workflow management: modeling concepts, architecture and implementation.* International Thomson Computer Press, 1996.
10. D. Jackson, *Software Abstractions: logic, language, and analysis.* MIT press, 2012.
11. H. Leopold, J. Mendling, and A. Polyvyanyy, "Supporting process model validation through natural language generation," *IEEE TSE*, vol. 40, no. 8, pp. 818–840, 2014.

[2] There is also a standalone version of MuDePS: `https://youtu.be/JhqSiAxChKQ`
[3] All tools and some test data are available at `http://mps.kppq.de`.