Studies on Declarative Process Modeling and Its Relation to Procedural Techniques (Extended Abstract)

Johannes De Smedt

The challenging task of managing business processes has become an even more complex endeavor as companies are required to sustain flexible and agile business practices. Business Process Management (BPM) research has proposed numerous ways of accommodating for this need for flexibility. The declarative process modeling paradigm, with its shifts towards a constraint-based way of approaching process behavior, is a prime example. By capturing a process' behavior in a loose fashion by demarcating what is- and is not allowed, the models produced by adhering to the paradigm are often more concise than their procedural counterparts. The downside of numerous languages covered by this paradigm, however, is the complex nature of their constraints, as well as their interactions. Furthermore, declarative process modeling might not offer a single golden bullet approach to attain flexibility, as it makes delineating structured process behavior often harder. Rather, an intermediate solution in the form of a mixed-paradigm model can provide a fitting answer that joins best of both worlds in areas with multiple layers of flexibility.

This thesis offers numerous solutions in the form of approaches, frameworks, and implementations to overcome these impediments for improved usability of declarative process modeling, as well as the comparison of both paradigms, and the in-depth study of mixed-paradigm process models. It does so in three major parts.

In the first part, the backdrop in terms of literature and formalisms is provided. Chapter 2 reports a comprehensive literature study that encompasses all works related to the declarative process modeling paradigm. It positions each of the works along the business process management lifecycle, hence assessing the maturity of this subfield of business process management. All papers published on the topic in the last 10 years were reviewed and classified according to their use of a certain phase of the lifecycle. Findings suggest that there is a strong skewness towards the discovery, implementation, and monitoring phases. This is mainly due to the formal nature of many papers, that focus on execution semantics, formal verification, and automated process discovery from event logs.

The key takeaways are summarized in a roadmap for future declarative pro-

cess modeling literature. First of all, it is suggested that a comparison framework for existing frameworks for scalability, applicability, and functionality can boost the focus of the research area. While one of the key benefits of using declarative process modeling is achieving flexibility, it remains to be seen how and how well each approach is capable of doing so. E.g., a study on how Declare with a bigger but unwieldy constraint base can compete with the more focused DCR Graphs approach in a real-life setting would offer more insight into how to progress the control-flow perspective. On the other hand, the integration of artifact-centric approaches delivers yet another view that rather focuses on integrating the lifecycle of objects. Secondly, modeling guidelines can make a welcoming addition to the field. Many studies have devoted their attention to the understandability of declarative process models, with often mixed to negative results. Modeling guidelines that are tailored towards the characteristics of declarative process models might alleviate this impediment, and is also an important aspect of this thesis in part 2. Thirdly, given the recent gain in popularity of decision modeling approaches which are also by nature declarative, it remains to be seen how a fully declarative solution of control flow, data, decision, and so on can unite in a process context.

In chapter 3, the formal layer of the thesis is provided. The concepts of activities, constraints, and models are introduced, along with the execution semantics of Petri nets, and of the widely-used Declare language in regular expressions, LTL, and R/I-nets. In the remained of the work, Declare will be used to illustrate the concepts that are studied and proposed.

In the second part of the thesis, the focus is shifted towards the understandability and usability of constraint-based declarative process models. Since often declarative process models are based on a vast number of interacting constraints, it becomes hard for modelers and users to grasp their interplay. Since the nature of the patterns is also of a wide variety, the effect each construct has on another is often obscured when creating bigger models. Especially the problem of hidden dependencies, i.e., interactions between constraints that are defined over the same set of activities that result in behavioral outcomes that are not covered by the separate semantics of constraints themselves, pose a significant threat to the inexperienced and even experienced process modeler. Therefore, in Chapter 4, an approach is devised that uncovers the interplay of Declare constraints.

Based on different types of constraints, it is possible to construct a recursive back- and forward searching procedure to establish so-called dependency structures. These structures visualize how constraints are related within a model, and what violations and resolutions have effect on the other ones. The approach is based on propagating the upper- and lower bound of the number of occurrences of each of the activities. Indeed, it is shown that the hidden dependencies stem from the constraints that impose an upper bound on activities, which consequently affect the activities connected to them.

In Chapter 5, the applications of revealing the dependencies are discussed.

Making the hidden dependencies explicit in dependency structures allows for the construction of an extra annotation layer of declarative process models in the form of textual descriptions of the dependencies. This was implemented in the Declare Execution Environment and used in a user experiment with a considerable amount of students. It was tested whether this extra layer and descriptions can aid novice modelers and users in understanding the behavior of declarative process models expressed in Declare. Next to this, also the annotation of the enabledness of activities and the violation status of constraints was tested. Results show that every layer, compared to a model with only the typical graphical notation of Declare without any other information but the separate constraint descriptions, significantly relieves mental effort as measured in the scores of the users, as well as the time needed to perform the questions regarding the models. Especially the fully-annotated models containing descriptions and dependency structures of the hidden dependencies renders novice users able to read and explain the behavior that is present in the models.

Following the user study, it is investigated how the insights on dependencies can be used towards quantifying the complexity of declarative, constraint-based process models. Traditionally, such measures in BPM focus on routing constructs and the number of paths that can be traversed in the process, i.e., the size of the language. In declarative models, however, there is little correspondence between the number of constraints and the number of paths because the effect of conjoining different constraints can lead to both an in- and decrease of the size of the language. By using the hierarchy that is inherent to the dependency structures for declarative process models, it is possible to follow a different route that is rather based on the fan-in, fan-out principle of tracking and tracing that also adheres to the open world assumption of declarative process models. The proposed complexity metric increases in effect size when there are more, denser, and more convoluted dependency structures present to reflect the difficult interactions that take place between the constraints.

Finally, Chapter 5 concludes with a preliminary approach to restructure declarative process models according to the dependency structures. By focusing on the constraints that cause hidden dependencies, and hence the structures, it is possible to derive different stages of the model. This allows to split the model and present it without hidden dependencies altogether.

In the third part, the comparison with the procedural process modeling paradigm is made for both manual, as well as automated process discovery and verification. In Chapter 6, an approach is devised for constructing mixedparadigm process models with intertwined state spaces, i.e., models consisting of both procedural and declarative process modeling constructs that share the same activities and are not just a collection of atomic subprocesses.

First, the different types of mixed-paradigm models are discussed. Such models can either incorporate declarative constraints for a certain smaller part of the behavior to increase flexibility, or for an extensive part of the model where declarative constructs can more concisely express flexible behavior. Furthermore, not only more flexibility can be achieved, as declarative constraints can also be especially restrictive. Hence, they can also tie down the behavior considerably with, e.g., constraints determining cardinalities for and constraints imposing chaining behavior on activities.

Next, all constraints in the Declare template base are reviewed for their impact on global concurrency, the timing and violation of constraints, and their effect towards permanently disabling activities. This is summarized in a scoring table that can be used to quantize to what extent they have an impact on a process model, and hence to what extent they might interfere with a procedural counterpart of a model.

Finally, the chapter concludes with a step-wise approach towards mixedparadigm process modeling. Four steps are put forward for users to construct better models. The approach is illustrated on a well-known example in the BPM sphere to display its usefulness and effect.

Chapter 7 starts from the same evaluation of how mixed-paradigm models can be useful, but on top also discusses the relation to the behavior in an event log. Again, declarative process constructs can be incorporated into procedural models by either being more restrictive, aiming for an even more precise result, and/or by covering more behavior in the log as it is capable of better capturing the flexible behavior.

The insights are condensed into the Fusion Miner framework, which offers an approach for mining mixed-paradigm models by using various procedural and declarative process mining algorithms. Its main idea starts from classifying activities into being either rather prone to be suitable in a procedural model, i.e., easy to capture within a structured process sequence, or being rather suitable for a declarative model, i.e., its place in a sequence is hard to express because of its looping behavior, random occurrence, or duplication. After this classification, each set of activities is mined accordingly with its respective paradigm. A userdefined parameter, the entropy level, controls the sensitivity towards regarding certain behavior as being fit for mining with declarative constraints. Both sets do have overlaps nonetheless, and it is illustrated how the declarative part of the model is capable of explaining the more volatile behavior generated by flexible processes better.

The approach is instantiated in this chapter in the form of a mixture of both Heuristics Miner and Declare Miner, and shows its ability towards mining better models in terms of fitness and precision.

Chapter 8 concludes the part of mixed-paradigm modeling, by providing a model verification and conformance checking approach. The former can be done by constructing the state space of the procedural part of the process, and subsequently conjoining the separate declarative constraints. Ultimately, either a global automaton is created containing all behavior that is modeled, or a tradeoff has to be achieved that preferably includes as many constraints in the model as possible. Also, an improved version of the Fusion Miner implementation is introduced, FusionMINERful, which uses both Inductive Miner and MINERful. The algorithm is also capable of finding the right level of entropy itself. Starting off by trying to build a reachability graph for the Petri net, it traverses the entropy spectrum when no such construct can be obtained, i.e., the share of the declarative activities in the overall model is increased. In case no deterministic procedural model can be built from the behavior witnessed in athe event log, a fully declarative model is returned.

To evaluate and compare procedural, declarative, and mixed-paradigm approaches, an alignment-based conformance checking technique was implemented that uses the global automaton of the model checking approach to replay traces. This allows for calculating fitness, precision, and generalization for the full spectrum of entropy.

Experiments were done on two synthetic, and one real-life event log. Results show that the intuition that procedural process models are better capable of representing more structured data than declarative models, and vice versa, is true, however, the gap in terms of performance is not considerable. Furthermore, mixed-paradigm approaches are often an as interesting, if not better solution, without needing that many more constructs. Although the evaluation of model understandability remains a subjective, the models produced by the algorithm seem to represent the behavior in an interesting way. Finally, the self-learning capabilities of the entropy level renders FusionMINERful a good tool to scrutinize the level of flexibility in an event log.

Finally, the last part provides an outlook for future work. Every part has an extensive line of further investigation that can be pursued. The topic of understandability can be further enriched by empirically validating the complexity metric. Also, a mature approach to restructure declarative process models along the phases can provide models that are easier to integrate with procedural models. The mixed-paradigm conformance checking study might benefit from benchmarking all procedural and declarative process models, and score them along the conformance dimensions to find the best mining solution overall. These are mere examples, and the range of future work is considerable.

To summarize, the thesis provides a plethora of techniques, approaches, and frameworks for improving the usability and understandability of constraintbased declarative process models and their integration with procedural models.