Declarative Modeling—An Academic Dream or the Future for BPM?

Hajo A. Reijers[1,2], Tijs Slaats[3,4], and Christian Stahl[1]

[1] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{H.A.Reijers, C.Stahl}@tue.nl
[2] Perceptive Software, Piet Joubertstraat 4, 7315 AV Apeldoorn, The Netherlands
[3] IT University of Copenhagen, Rued Langgaardsvej 7, 2300 Copenhagen, Denmark
TSlaats@itu.dk
[4] Exformatics A/S, Lautrupsgade 13, 2100 Copenhagen, Denmark

Abstract. Declarative modeling has attracted much attention over the last years, resulting in the development of several academic declarative modeling techniques and tools. The absence of empirical evaluations on their use and usefulness, however, raises the question whether practitioners are attracted to using those techniques. In this paper, we present a study on *what practitioners think of declarative modeling*. We show that the practitioners we involved in this study are receptive to the idea of a *hybrid approach* combining imperative and declarative techniques, rather than making a full shift from the imperative to the declarative paradigm. Moreover, we report on requirements, use cases, limitations, and tool support of such a hybrid approach. Based on the gained insight, we propose a *research agenda* for the development of this novel modeling approach.

1 Introduction

Imperative modeling is currently the most prominent modeling paradigm in BPM. Imperative modeling techniques are implemented in almost every modeling tool, and many imperative modeling languages have been developed, most prominently, Event-Driven Process Chains (EPCs) and Business Process Modeling Notation. Imperative models take an *"inside-out"* approach; that is, every possible execution sequence must be modeled explicitly. As a consequence, imperative modeling may lead to over-specification and lack of flexibility, making it difficult to defer decisions at runtime and to change existing process models [21,2].

To overcome these shortcomings, *declarative modeling* approaches have been proposed [3]. In contrast to imperative approaches, declarative models take an *"outside-in"* approach. Instead of describing how the process has to work exactly, only the essential characteristics are described. To this end, constraints are specified that restrict the possible execution of activities.

Research on declarative modeling has gained increasing interest over the last years. Declarative languages, such as Declare [3] (formerly known as DecSerFlow), DCR Graphs [12] and SCIFF [14], have been developed. These languages have been integrated in academic and industrial modeling tools [24].

Beside the development of declarative techniques, also empirical research has been conducted to study the relation between imperative and declarative approaches [8,9,22,20]. It is well understood how to specify properties of a business process, but it is still not clear how to define a business process modeling languages that is understandable [8] on the one hand, and enables maintainability [9], expressiveness and modeling comfort, on the other hand.

To the best of our knowledge, there does not exist any studies that reflect on the question whether declarative techniques can be used in practise from a practitioner's standpoint. This raises a question, which has not been answered yet: *Do practitioners see opportunities to use declarative techniques?*

The contribution of this paper is to present *what practitioners think of declarative modeling*. In that way, we close the gap between research on declarative techniques and empirical investigations on declarative modeling. Our results are based on a workshop on declarative modeling with ten professionals from industry, including both consultants involved in modeling projects and developers of industrial modeling tools. During the workshop, we introduced declarative modeling techniques, performed two modeling assignments, and discussed the prospects of a declarative approach. The evaluation, both qualitative and quantitative, shows that practitioners see good opportunities for a *hybrid approach combining imperative and declarative techniques* while they are skeptical regarding a purely declarative approach. With the gained insight from the discussion, we present requirements on such a hybrid approach, use cases, limitations, and requirements concerning tool support. Shifting the focus from imperative and declarative modeling to a hybrid approach raises many research questions. Therefore, we propose a *research agenda* for the BPM community to make the hybrid approach work.

We continue with a brief introduction to Declare and DCR Graphs, two declarative approaches we used throughout the workshop. In Sect. 3, we describe the outline of the workshop and our evaluation method. The quantitative evaluation is described in Sect. 4, and Sect. 5 reports on the qualitative evaluation. In Sect. 6, we present our research agenda. We close with a conclusion and directions for future work.

2 Declare and DCR Graphs by Example

In this section, we briefly introduce two declarative modeling approaches, Declare [3,24] and DCR Graphs [12], using the following example of a document management system. To simplify the presentation, we restrict ourselves to the control flow dimension and do not consider data or resources.

Example 1. Every case of the document management system is initially *created* and eventually *closed*. For a created case, an arbitrary number of documents can

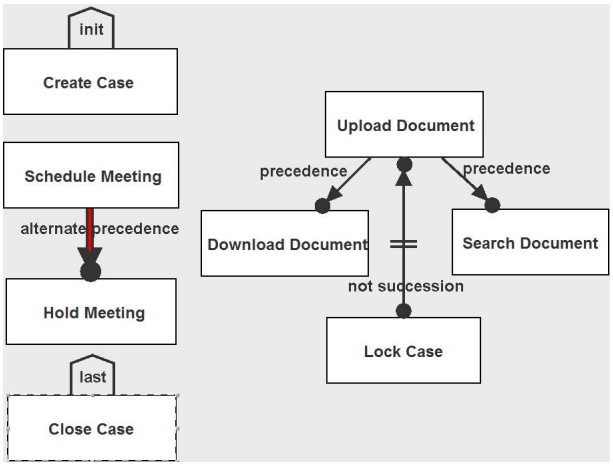

Fig. 1. Declare model of the document management system

be *uploaded*. An uploaded document can be *downloaded* or *searched*. At any time, a case can be *locked*. After locking a case, it is not possible to upload a document; still, uploaded documents can be downloaded and searched. Furthermore, in every case, meetings can be *held*. To hold a meeting, it has to be *(re-)scheduled*. Meetings can be rescheduled arbitrarily often, but it is not possible to schedule more than one meeting in advance.

2.1 Declare

A Declare model consists of activities and constraints. An activity is depicted as a rectangle and a constraint as a hyper-arc (i.e., a constraint connects one or more activities). From the specification, we identify eight activities which are highlighted in the description. Figure 1 shows the Declare model of the example. The *init* symbol on top of activity *Create Case* specifies that every case of the document management system starts with activity *Create Case*. Likewise, the *last* symbol on top of activity *Close Case* specifies that the final activity of every case of the document management system is *Close Case*.

There are three types of arcs in Fig. 1. Each arc type specifies one type of constraint. The precedence constraint, modeled as an arc from *Upload Document* to *Download Document* specifies that a document has to be uploaded before it can be downloaded. Likewise, we can only search a document once it has been uploaded (arc from *Upload Document* to *Search Document*). The second type of constraint is the not-succession constraint, which is modeled by an arc from *Lock Case* to *Upload Document*. It specifies that after a case has been locked, we cannot upload new documents. The third type of constraint, alternate precedence, is the arc from *Schedule Meeting* to *Hold Meeting*. It means that a meeting can only be held after it has been (re-)scheduled at least once. Moreover, after a meeting has been held, the next meeting has to be (re-)scheduled before

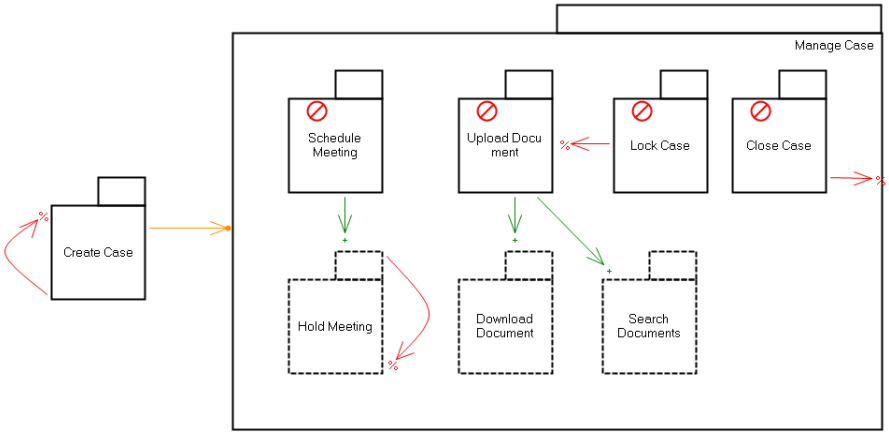

Fig. 2. DCR Graph model of the document management system

it can be held (i.e., activity *Hold Meeting* has to be followed by *Schedule Meeting* before *Hold Meeting* can be executed again).

As mentioned in the introduction, a declarative model only describes the essential characteristics of a process rather than how the process has to work exactly. For example, holding and (re-)scheduling meetings is independent from handling documents. Therefore the respective activities are not connected by arcs; that is, no constraint restricts their interplay. To execute the model in Fig. 1, one has to determine which activities are enabled by evaluating all constraints. Initially, it is the start activity, *Create Case*. After this activity is executed, any of the activities *Schedule Meeting*, *Upload Document*, *Lock Case* and *Close Case* can occur. A Declare model can be enacted and executed. The tool then computes the enabled transitions for every state [24].

2.2 DCR Graphs

A DCR Graph model consists of activities, relations, and a runtime marking. Activities are depicted as rectangles with an "ear" that can contain the roles which can execute the activity. Activities can be nested under super-activities, depicted by drawing an activity inside the rectangle of another activity, in which case any relation that applies to the super-activity, applies to all its sub-activities. Only the atomic activities (that do not contain any sub-activities of their own) are executable. The relations are drawn as arrows between activities.

Figure 2 shows the DCR model of the example. The first activity is *Create Case*, which should occur before all other activities can occur. We model this behaviour by the yellow *condition relation* from *Create Case* to the super-activity *Manage Case*, containing all other activities. The condition relation states that the second activity (in this case any sub-activity of *Manage Case*) can not occur before the first activity (in this case *Create Case*). We also require that *Create*

Case happens only once, which we model through the *dynamic exclusion relation* drawn as a red arrow with a percentage sign at the end. Through this relation *Create Case* excludes itself from the workflow when it is executed, meaning that it can not be executed anymore afterward. The next two activities are *Schedule Meeting* and *Hold Meeting*. We should always schedule a meeting before we can hold a meeting, but it might be the case that a meeting is rescheduled before it is held. We model this in the following way: *Hold Meeting* is *initially excluded*, meaning that at the start of the workflow it can not be executed before it is included. *Hold Meeting* is included by doing *Schedule Meeting*, modelled by the *dynamic inclusion relation*, drawn as a green arrow with a plus sign in the end. *Hold Meeting* excludes itself meaning that it can not be executed again before there has been a new occurrence of *Schedule Meeting*. The next three activities are *Upload Document*, *Download Document*, and *Search Documents*. We can not download or search documents before at least one document has been uploaded, therefore those activities are initially excluded and will be included by *Upload Document*. The case can also be locked through the activity *Lock Case*, which makes it impossible to upload further documents, therefore *Lock Case* excludes *Upload Document*. Finally we can close the case by executing the activity *Close Case*. We model this by having *Close Case* exclude the super-activity *Manage Case*. Because all activities are nested under *Manage Case*, *Close Case* will exclude all activities from the workflow.

The final two relations of DCR Graphs are not used in the example. First there is the response relation which states that one activity requires another activity to happen in the future, when this occurs we say that the second activity is a *pending response* and annotate it with an exclamation mark. A workflow is in an accepting state while there are no *included pending responses*, in case there are included pending responses these should be executed before the workflow can be closed. The second relation that is not shown is the milestone relation, it captures this accepting condition on the level of activities by stating that while some activity is a pending response, some other activity can not be executed.

We represent the runtime of a DCR Graph by showing which activities have been executed at least once before by drawing them with a green check-mark, showing which activities are pending responses by drawing them with a red exclamation mark and showing which activities are currently excluded by drawing them with a dashed line instead of a solid line. We call these three sets of activities the *marking* of the DCR Graph. Based on the marking we can determine which activities are enabled: Activities which are excluded (drawn with a dashed line) are not enabled and activities that are blocked by a condition and/or milestone relation are also not enabled. In the latter case, we show this by drawing a red stop-mark on the activity. In Fig. 2, one can see that the only initially enabled activity is *Create Case*. All other activities are either excluded (drawn with dashed lines) or blocked through the condition relation (drawn with a red stop-mark). We distinguish between being excluded and blocked by a condition/milestone relation because we consider these two as essentially different states of the activity: When it is blocked it is still a part of the workflow, but

being stopped from executing. When it is excluded it is not considered as a part of the workflow at that time. This is also why only *included* pending responses will block the workflow from being closed.

2.3 Comparison

Figures 1 and 2 clearly illustrate the idea behind declarative modeling. The main difference between DCR Graphs and Declare is that the DCR Graph approach allows to define any constraint using the five basic relations, while one has to define many more constraint for Declare (some of them are logical combinations of simpler constraints). Also, Declare represents the runtime of a workflow by showing the state of the individual constraints—that is, which constraints are (possibly) satisfied, and which constraints are (possibly) violated. DCR Graphs, by contrast, represents the runtime of a workflow by showing which tasks have been executed at least once before, which tasks are pending as a response and should be done some time in the future, and which tasks are currently included in the workflow. While on infinite traces DCR Graphs are strictly more expressive than Declare, this has no impact on practical business process modeling: The processes under consideration typically produce finite traces.

3 Method

For our evaluation, we worked together with Perceptive Software, a provider of enterprise content management and BPM technologies. We invited both consultants, who engage with clients to model their processes and implement BPM suites using such models, and professionals who contribute to the development of the toolsets. We planned a single workshop that went through the four phases, as depicted in Fig. 3.

In the first phase (*Introduction*), we provided the participants with the motivation for organizing the workshop and gave them a generic introduction to declarative principles. After this phase, we split the groups into two sub-groups of equal size. We first randomly assigned half of the consultants to group 1 and the other half to group 2. We did the same for the developers after that. In this way, we ensured an even distribution of consultants and developers over the groups.

The second phase was specific for each group and consisted of a tutorial on the techniques under consideration (*Explanation*). In other words, one group received the tutorial on DCR Graphs and the other on Declare. The tutorials were provided by separate moderators for each group. Each moderator had deep expertise in the technique that he explained. The tutorials were synchronized beforehand between the moderators to guarantee a similar level of depth and the same duration.

Following up on the tutorials, each group received two assignments. These assignments were the same for both groups and required the participants to translate the assignment material into process models (*Modeling*). Clearly, the

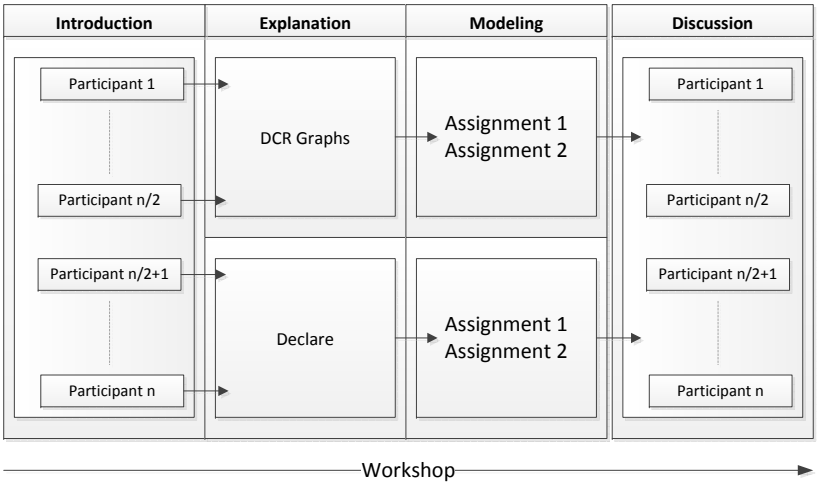

Fig. 3. Organization of the workshop

sub-group who received the tutorial on DCR Graphs used this technique; the other sub-group used Declare. The assignments can be found back at `http://www.win.tue.nl/ais/doku.php?id=research:declare:bpm2013`. As we were not so much interested in checking the correctness of the solutions but in transferring knowledge on the techniques to the participants, we encouraged them to work in pairs within each sub-group.

The final phase re-united the sub-groups (*Discussion*). During this phase, we first had the participants fill out a questionnaire on usefulness, ease of use, and intent to use as proposed by Moody [17]. The questionnaire can be used to get a broad-brush insight into the perceived quality of an IS design method, building on the concepts known from the Technology Acceptance Model as proposed by Davis [6]. We extended the questions with some more to gather demographic data on the group. The used questions can also be found at `http://www.win.tue.nl/ais/doku.php?id=research:declare:bpm2013`. After the questionnaire, we engaged in a semi-structured discussion with the group. This discussion was moderated by one of the authors, while the other authors took notes. The independently taken notes were used to reach consensus on how the participants reflected on the questions.

The insights that we gathered during the last phase of the evaluation with the questionnaire will be referred to as the *quantitative evaluation*, because the design of Davis' list allows for measuring the strength of the perceptions on ease of use, usefulness, and intent to use. Our insights on the modeling phase and the open part of the discussion phase will be dealt with as the *qualitative evaluation*, as they add a qualifying lens on the results. These respective evaluations will be discussed next.

4 Quantitative Evaluation

4.1 Demographics

Overall, ten professionals participated in the workshop. Of these, five are active as consultants, modeling processes at client sites and implementing process management software, while the other five are involved in different roles associated to the development of process modeling and workflow tools (product manager/architect/developer). For the entire group, the average number of years of experience in the BPM domain was more than 11 years. Of the ten participants, on a scale of 1 to 5, three considered themselves to have an intermediate expertise in process modeling (level=3), three to have an advanced level of expertise (level=4), and the remaining four people considered themselves to be experts (level=5). Finally, the participants indicated that on average they had each read close to 15 different process models in the preceding 12 months, while each had created or updated nearly four models on average in the same period. We are aware that the number of professionals is rather low. However, within a given time frame, we were choosing the day for which most professionals indicated their availability.

4.2 Validity and Reliability

Prior to performing an in-depth analysis of the data that had been gathered through the questionnaire, the validity and reliability of the empirical indicators were checked. We determined all correlations between the responses for questions that were used to measure to same construct (inter-item correlations) and identified no item that displayed a low convergent validity. In other words, the questions and their grouping to measure the constructs appeared valid. Next, we used Cronbach's alpha to test the reliability of the items to measure the various constructs. This is a test to check internal consistency of the questions. While there is no authoritative level for Cronbach's alpha, it is generally assumed that levels above 0.7 point at a good reliability of the items [18]. Adequate levels were established for *Perceived Usefulness* (0.743) and *Perceived Ease of Use* (0.826). However, *Intention to Use* scored too low (0.600). For this reason, we removed the latter construct from our main analysis and will only report on the mean scores of the items. Note that it was the only construct measured using just two items—an approach to be reconsidered in future applications of the questionnaire.

4.3 Results

Our main analysis then focused on this question: Are the considered techniques, DCR Graphs vs. Declare, perceived differently by the groups? To select the appropriate technique, we established with the Shapiro-Wilk test that the respondent answers were normally distributed. We could, therefore, proceed with applying a one-way ANOVA test with Perceived Usefulness and Perceived Ease

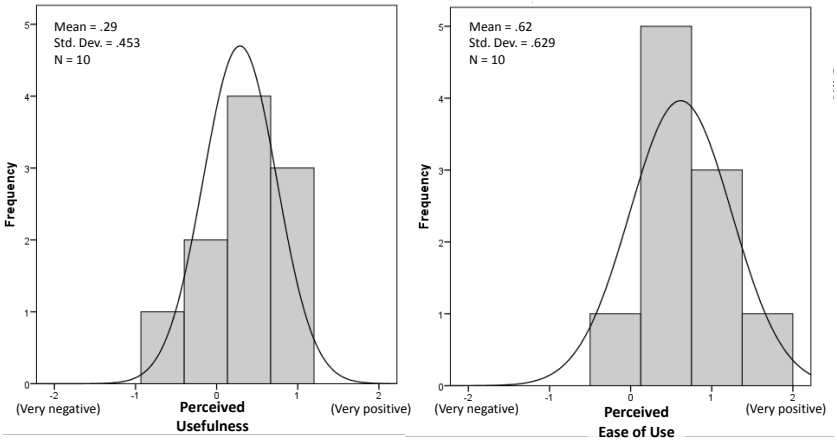

Fig. 4. Histograms for Perceived Usefulness and Perceived Ease of Use

of Use as dependent variables and the technique employed as factor. The test generated p-values of 0.116 and 0.939 for Perceived Usefulness and Perceived Ease of Use, respectively. By maintaining a confidence level of 95%, both of these values exceed the 0.05 threshold. In other words, any differences in perception between the used techniques are not statistically significant. Therefore, we must reject the idea that people perceive the techniques as different in either their usefulness or their ease of use.

This first important insight allows us to aggregate the responses received from both groups to determine a view on the usefulness and ease of use of declarative techniques on a more general level. Figure 4 shows the histograms for the two constructs under consideration, Perceived Usefulness and Perceived Ease of Use, aggregating the responses from all ten respondents. Also displayed is the fitted normal distribution for both constructs.

The histograms display the frequencies of the scores on a scale of -2.0 (very negative) to +2.0 (very positive). The 0 value indicates the neutral stance (not negative, not positive). What can be seen is that the averages of the distributions for both constructs are positive, hinting at a receptive mood toward declarative techniques in terms of both constructs. Note that the mean values for the two items under consideration for Intention to Use are 0.00 and 1.00. Second, Perceived Ease of Use seems to be more positively evaluated than Perceived Usefulness, with respective mean values of 0.62 and 0.29.

To determine whether the optically favorable outcomes are indeed statistically significant, we applied one sample t-tests. Like in our previous test, we used a confidence level of 95%, which means that we will only treat p-values below 0.05 as statistically significant. The outcomes of the t-tests are that the positive mean score for Perceived Ease of Use is significantly different from zero (p=0.013), but that this is—just—not the case for Perceived Usefulness (p=0.076). In other words, one can trust that the positive stance toward the ease of use is not a

matter of chance. However, this cannot be ruled out for usefulness, despite its closeness to the cut-off value. Apparently, the involved respondents can easily use the method, despite the limited amount of training received. They were not similarly outspoken about the usefulness of a declarative technique, albeit certainly not negative either.

We finally checked whether the years of experience, the level of expertise, the type of role (consultant vs. non-consultant), or the modeling intensity in terms of models read or created had any relation to the outcomes. Interestingly, we could see that the most negative responses on Perceived Usefulness came from those respondents who assessed their own level of process modeling expertise as relatively low. While on average the three respondents with an intermediate expertise assessed the usefulness of the declarative techniques as negative (-0.208), the advanced modelers and the experts were positive (0.417 and 0.563, respectively). Tukey's HSD (Honestly Significant Difference) confirmed that the self-assessed level of expertise was a significant factor to explain differences in scores on Perceived Usefulness ($p=0.042$). In other words, the higher the level of expertise, the more merit a participant saw in the declarative techniques. The other factors had no noticeable effects on the scores.

5 Qualitative Evaluation

In this section, we present the qualitative evaluation of the workshop. In particular, we report on the results of the modeling assignment and of the discussion with the professionals.

5.1 Modeling Assignment

As reported in Sect. 3, we split the ten professionals into two groups of five. One group got an introduction to Declare and the other group to DCR Graphs. After this introduction of about 30 minutes, each group was asked to work on two small modeling assignments. One assignment was the document management system, which we used to illustrate Declare and DCR Graphs in Sect. 2. The second assignment was a hospital process of similar size and level of difficulty. The professionals worked in groups of two and three on the two assignments. Each assignment took less than 15 minutes, after which we presented and discussed our solution. All four groups came up with a correct solution for each of the two assignments.

The way we organized the assignment does not allow us to derive overly strong conclusions. Still, we gained two interesting insights. First, the result of the assignment shows that it is possible to teach declarative modeling to practitioners. Although it was difficult for the professionals to get used to the declarative way of modeling and to the graphical notations of the techniques, they came up with correct models in reasonable time. Second, we were told that the graphical notation of Declare and DCR Graphs are too academic and for practitioners neither convincing nor intuitive. Moreover, also the informal

description, which we provided for each introduced constraint, did not help them to easily identify the constraint they needed. These comments hold for both techniques, DCR Graphs and Declare. This comes, indeed, not as a surprise as both formalisms have an academic background. However, we expected DCR Graphs to be more comprehensible and easier to use than Declare because DCR Graphs only consist of five relations, whereas Declare requires to learn a larger set of constraints.

5.2 Opportunities for a Declarative Approach

In the subsequent discussion with the professionals, we tried to figure out whether they see opportunities for a declarative modeling approach. Clearly, such a question is difficult to answer given the the short tutorials and only taking two assignments. The participants did indicate that there are probably processes that can be modeled most naturally using the imperative approach, while others would fit better with the declarative approach. For example, a clearly well-structured process of registering a newborn at a townhall can be modeled most naturally in an imperative way whereas the document management system of Sect. 2 is an example of a process that can be modeled most natural in a declarative way. In addition, in almost all processes the professionals came across, there were always at least parts or subprocesses where an imperative approach seems most natural. So, the conclusion to this question is that a purely declarative approach seems less attractive than a *hybrid approach*, which combines imperative and declarative modeling aspects.

5.3 Requirements Concerning a Declarative/Hybrid Approach

In the previous section, we showed that practitioners see opportunities for a hybrid approach. Next, we report on the practitioners' requirements concerning the specification, the constraints, the process, and tool support.

The consensus was that the efficient design of a declarative model (or of the declarative part of a hybrid model) will require a declarative specification. The reason is that it can be nontrivial to derive constraints from an imperative specification. We received one comment that it might be difficult to get a declarative specification at all, but we are not that pessimistic. Based on our experience, it depends on how one formulates questions to domain experts; that is, asking about the relationship between two activities (i.e., declarative) rather than which steps can be performed in a certain state (i.e., imperative) will allow one to come up with a declarative specification.

Other requirements concern the constraints. The involved professionals brought forward that too many constraints may negatively influence the quality of a declarative model. For example, many constraints affecting few activities may result in an unreadable model. Furthermore, they assumed the completeness of the constraints to be crucial, although that is similar to the completeness of the branching conditions in an imperative model.

Looking at the process model or the specification to identify "candidate" parts that may benefit from a declarative modeling approach, the professionals suggested to identify parts that have many dependencies (e.g., spaghetti-like parts). Although such parts seem good candidates, it is unclear whether a declarative way of modeling results in a better model. Another suggestion was to identify those parts where much modeling freedom is; for example, a set of concurrent activities (that preferably occur more than once) with only few dependencies may result in a simpler declarative model than their imperative counterpart.

Finally, also proper tool support is a hard requirement. Here, in particular, the professionals saw deficits in the usability of the academic techniques when used in a pen-and-paper fashion. We shall discuss tool support in Sect. 5.6 in more detail.

5.4 Use Cases for a Declarative/Hybrid Approach

In this section, we list use cases for the declarative approach as identified during the discussion.

Process evolution [21] was mentioned as the main use case by the involved professionals—that is, manage processes along the various changes it encounters. Having a set of constraints rather than a graph-based model seems to be beneficial to visualize changes over time, on the one hand and to actually change a process model, on the other hand. This is, in fact, one of the claimed advantages of declarative modeling [3,15]. The "outside-in" approach of declarative models allows for a higher level of abstraction than an imperative process model. Therefore, it is often simpler to add or remove constraints than changing a BPMN model, for instance.

The discussion also suggested that the use case for declarative models is tied to model purpose. Process models serve different purposes—for example, as a medium to communicate with stakeholders (i.e., communication model) or to execute a process (i.e., executable model). Especially with respect to the communication aspect, the professionals saw good opportunities for using declarative techniques. Communication models are rather imprecise (e.g., exceptions may be left out), and business analysts do not tend to stick to model conventions. Instead, they may prefer to use short hands to illustrate behavior in a simplified way, for instance. Here, a hybrid approach looks promising as the business analyst is provided with a lot of different ways to present the model. Again, this follows from the higher level of abstraction of declarative models. In contrast, there was no common agreement on a declarative approach being useful for specifying executable models. As an executable model contains all behavior, a hybrid approach will only be beneficial if it allows for designing more readable or simpler models.

Another interesting aspect mentioned was that a hybrid approach may result in fewer errors in the model than using a purely imperative approach. This may lead to shorter development cycles. We think that this is also a consequence of the higher level of abstraction in declarative modeling. A modeler has to

identify the constraints rather than encode it in terms of control flow. However, no experience report or empirical results exist that confirm this assumption.

5.5 Limitations of a Declarative/Hybrid Approach

In this section, we report on limitations of a declarative/hybrid approach concerning the specification, the modeling paradigm, and the usability.

The main concern regarding the specification is that currently all specifications are imperative (e.g., "we first do this, then that"), and it seems to be very difficult to produce a declarative model for such a specification. As discussed in Sect. 5.3, we think that it is possible to receive declarative specifications.

There has been a paradigm shift in system development from monolithic systems to component-based systems that are distributed within and across organizational boundaries. One prominent computing paradigm that implements this trend is service-oriented computing (SOC) [19]. We received concerns that in this setting declarative modeling techniques may be less applicable compared to imperative techniques. The reason for this concern lies in the fact that certain constraints affect activities of an individual component, whereas other constraints affect activities of different components. Declarative techniques have, however, been successfully applied in the service-oriented setting [14,15], and it has been studied how a declarative cross-organizational workflow containing global constraints can be projected to its individual localized components [11], so we are convinced that this concern is unsubstantiated.

Another limitation concerns the usability of existing techniques and tools. Current tool support is mainly academic by nature and seems, therefore, not overly concerned with usability issues. Moreover, the declarative paradigm also requires a different way of thinking, making it perhaps difficult for practitioners to understand declarative models. Here, more research is required to make declarative techniques more comprehensible.

5.6 Requirements Concerning Tool Support

In this section, we report on feedback we received concerning Declare and DCR Graphs and general requirements concerning tool support.

Several requirements on tools that were discussed deal with the specification and visualization of *constraints*. As mentioned earlier in Sect. 5.1, the professionals mentioned that working with constraints was relatively difficult for them. The graphical notations used in Declare and DCR Graphs were not always that intuitive. Moreover, specifying constraints in plain English is not always helpful either, because it is often nontrivial to identify the differences between two constraints. Therefore, the professionals proposed that constraints should be automatically derived from an informal textual specification. This problem has indeed been investigated in the field of computer-aided verification, for instance. Different approaches have been proposed, for example [7,5], but none of them could solve the problem entirely.

A given set of constraints makes it necessary to check for conflicting constraints. This is a feature which has been implemented in most declarative modeling tools [24], but has also been investigated in the context of compliance rules [4], for instance. Another important feature is to generate a model from a given set of constraints and to identify missing features. This problems is related to scenario-based programming [10]. In case an implementation and recorded event logs exist, process mining techniques to automatically derive missing constraints from the logs are required. First attempts at dealing with this topic exist, see [13].

Besides modeling support, tools should preferably also provide operational support. For example, event logs may be exploited to provide at runtime the best possible next step. Such features are implemented in recommender systems.

Finally, usability plays an important role. Specification of constraints, their graphical representation and the complete interplay between the tool and an end user must be on an abstraction level that is adequate to the task at hand.

6 Research Agenda

In this section, we pick up the results from the discussion with the professionals as presented in the previous section. We propose a research agenda for the development of a hybrid modeling approach that combines imperative and declarative techniques. The aim is thereby to point out necessary steps for developing and actually using a hybrid technique rather than a complete research agenda.

Model guidelines In order to apply the hybrid approach, a modeler has to know when to model in an imperative and when in a declarative way. In other words, we need to identify modeling guidelines to guide modelers through the modeling process. This requires rules for identifying imperative and declarative "candidate" parts on the level of an existing (imperative) process model (e.g., for process redesign), on the level of event logs (e.g., for process discovery), and on the level of (informal) specifications (e.g., for designing a new model).

Identify the hybrid technique Modeling in a hybrid way requires a well-suited modeling language. It needs to be investigated whether we can combine existing imperative and declarative languages or whether a new language has to be designed. For instance, we can integrate a declarative part as a subprocess into an imperative model (e.g., as a hierarchical transition in a Petri net or subprocess task in BPMN) or we can allow declarative and imperative constructs to coexist within a single subprocess. The modeling language must in any case support hierarchy. In the latest version of CPN Tools [23], Westergaard integrated DCR Graphs and Declare into Colored Petri nets. It turned out that defining the semantics of such models is nontrivial.

Beside that, it needs to be settled which constraints are relevant for practise and, thus, what the expressiveness of the declarative part of the language is. Empirical research has shown [8,9] that certain declarative constructs may be

more difficult to understand. Thus, we think the language should not contain too many declarative constructs, but this needs further empirical investigation.

Also, the graphical representation of hybrid models must be investigated. Different graphical notations exist, for example, compare DCR Graphs and Declare. Insights from [16] may aid the design of a hybrid notation.

Analysis of hybrid models The novel modeling approach needs analysis techniques including the verification of models, performance analysis, and property-preserving abstraction and refinement techniques. Also, process mining techniques [1] are needed—for example, checking the conformance of an event log and a hybrid model and discovering a hybrid model from a given event log.

Tool support To show the applicability of the hybrid modeling technique, tool support is a *sine qua no*. As reported in Sect. 5.6, research has to be performed to simplify the use of declarative techniques, for example, finding a way to derive constraints from informal specifications that can be used by business analysts without requiring knowledge about temporal logics.

7 Conclusion

We reported on a workshop on declarative modeling given to professionals from industry. The goal of this workshop was to gain insight into what practitioners think about declarative modeling and what opportunities they see to use this technique. Our quantitative evaluation showed that they were mostly positive and open to this modeling paradigm. In particular, the techniques were rather easy to learn. The qualitative evaluation showed that the practitioners did single out the use of declarative techniques in the context of a hybrid approach, which combines imperative and declarative modeling. Although our study is only based on a small group of practitioners, we are convinced that practise can benefit from such a hybrid modeling approach. To arrive at such an approach, we proposed a research agenda for the development of a hybrid approach.

In our ongoing research, we plan to work on the development of modeling guidelines. We will investigate techniques to identify "candidate" parts of a model for which a declarative way of modeling seems most natural. Also, we plan to study event logs and process models and try to use the results to identify constraints that frequently occur. In a second branch of research, we will investigate what a hybrid technique may look like, thereby using Declare, DCR Graphs and CPN Tools as starting points for our studies.

References

1. Aalst, W.M.P.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Aalst, W.M.P.v.d.: Business process management: A comprehensive survey. ISRN Software Engineering 2013 (2013)

3. Aalst, W.M.P.v.d., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009)
4. Awad, A., Weidlich, M., Weske, M.: Consistency checking of compliance rules. In: BIS 2010. LNBIP, vol. 47, pp. 106–118 (2010)
5. Cobleigh, R.L., Avrunin, G.S., Clarke, L.A.: User guidance for creating precise and accessible property specifications. In: SIGSOFT FSE. pp. 208–218. ACM (2006)
6. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. 13(3), 319–340 (1989)
7. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE 1999. pp. 411–420. ACM (1999)
8. Fahland, D., Lübke, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of understandability. In: BMMDS. LNBIP, vol. 29, pp. 353–366. Springer (2009)
9. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of maintainability. In: BPM Workshops. LNBIP, vol. 43, pp. 477–488. Springer (2010)
10. Harel, D.: Come, let's play - scenario-based programming using LSCs and the play-engine. Springer (2003)
11. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Safe distribution of declarative processes. In: SEFM 2011. pp. 237–252. Springer (2011)
12. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: PLACES 2010. EPTCS, vol. 69, pp. 59–73 (2010)
13. Maggi, F.M., Bose, R.P.J.C., Aalst, W.M.P.v.d.: Efficient discovery of understandable declarative process models from event logs. In: CAiSE 2012. LNCS, vol. 7328, pp. 270–285. Springer (2012)
14. Montali, M.: Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach, LNBIP, vol. 56. Springer (2010)
15. Montali, M., Pesic, M., Aalst, W.M.P.v.d., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. TWEB 4(1) (2010)
16. Moody, D.: The physics of notations: toward a scientific basis for constructing visual notations in software engineering. Software Engineering, IEEE Transactions on 35(6), 756–779 (2009)
17. Moody, D.L.: The method evaluation model: a theoretical model for validating information systems design methods. In: ECIS 2003. pp. 1327–1336 (2003)
18. Nunnally, J.: C.(1978). Psychometric theory. New York: McGraw-Hill (1978)
19. Papazoglou, M.P.: Web Services - Principles and Technology. Prentice Hall (2008)
20. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: An empirical investigation. In: BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 383–394. Springer (2012)
21. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems. Springer (2012)
22. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The declarative approach to business process execution: An empirical test. In: CAiSE 2009. LNCS, vol. 5565, pp. 470–485. Springer (2009)
23. Westergaard, M.: CPN Tools 4: Multi-formalism and extensibility. In: Petri Nets 2013. LNCS, Springer (2013), accepted for publication
24. Westergaard, M., Maggi, F.M.: Declare: A tool suite for declarative workflow modeling and enactment. In: BPM (Demos) 2011. CEUR Workshop Proceedings, vol. 820. CEUR-WS.org (2011)