Teaching Declarative BPM: Where, When and to Whom it Matters

Hugo A. López[1], Claudio Di Ciccio[2], Thomas Hildebrandt[3], and Johannes De Smedt[4]

[1] Technical University of Denmark, Kongens Lyngby, Denmark, hulo@dtu.dk
[2] University of Utrecht, Utrecht, The Netherlands, c.diciccio@uu.nl
[3] University of Copenhagen, Copenhagen, Denmark, hilde@di.ku.dk
[4] KU Leuven, Leuven, Belgium, johannes.desmedt@kuleuven.be

Abstract. The theory of BPM has, for a long time, discussed the interaction between flows and business rules, in what some call the *declarative vs imperative business process spectrum*. While imperative models constitute the standard in the curricula, with standards such as BPMN being imparted in most educations, declarative BPM has received a much more specialized scope. This paper collects and discusses insights coming from experiences teaching declarative process models in academia, in courses ranging from computer science to business informatics across universities in Austria, Belgium, Denmark, and the Netherlands, with languages such as Declare, DCR graphs, and DMN. We report considerations in curriculum development, considerations for diverse backgrounds, accessibility of tools and materials, and overall student perceptions of declarative modelling. Moreover, it reports some of the challenges of teaching declarative models in the classroom and outlines possible mitigation aspects that could support an easier adoption by the students.

Keywords: Declarative BPM, Declare, DCR graphs, DMN, Education in BPM

1 Introduction

The introduction of the declarative paradigm in Business Process Management (BPM) has given rise to various modeling languages and approaches for constructing, deriving, and improving them to facilitate the various stages of the BPM lifecycle [21]. Declarative Business Process Management (DBPM) is an approach within BPM for the modeling, execution, and mining of business processes that focuses on describing the constraints that processes must fulfill, rather than prescribing how activities should be ordered. Unlike classical imperative process modeling languages (e.g., workflow nets, BPMN), which define explicit control-flow sequences, declarative notations specify activities and a set of constraints governing the causal dependencies between them. These rules can impose control-flow as well as temporal and data-driven requirements. Any execution that satisfies all constraints is considered valid. This paradigm is particularly suitable for domains characterized by high variability, knowledge-intensive tasks, and frequent exceptions, such as healthcare, compliance, and case management [18,53]. However, the languages, approaches, and tools supporting DBPM require a different mindset, which also translates into the need for adapted declarative BPM education.

Over the years, the declarative paradigm has found its way into the BPM curriculum at various institutions, from teaching Dynamic Condition Response (DCR) Graphs and

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-032-36710-5_21

DECLARE for control flow and case-based modeling, to mining Decision Model and Notation (DMN) models to teach process redesign. Yet, they face the struggle of factors such as a maturing, often scattered technological landscape, as well as the inherent cognitive complexities that declarative modeling induces in learners. An initial set of challenges and research agenda was outlined in [54]. Now, more than a decade later, we would like to provide an account of how the educational aspects of DBPM have matured over the years and the remaining aspects to be supported in the future. In this way, this paper provides an overview of some of the efforts to integrate DBPM into BPM education. We collected experiences and answers regarding the ways DBPM has been implemented by instructors over the years from four European countries, in different educational contexts and adopting diverse declarative languages. We use these experiences to identify common themes and trends, which were agreed upon by the author team. As a result, we outline the main challenges in educating DBPM and propose a set of educational and research challenges to better integrate declarative approaches into BPM education.

The rest of this paper is structured as follows: Sect. 2 introduces the declarative paradigm for BPM; Sect. 3 outlines experience gained from teaching DBPM and its role in BPM; Sect. 4 discusses extant challenges in education related to DBPM; finally, Sect. 5 concludes the paper with an agenda for the further development of DBPM curricula.

2 Declarative BPM

To illustrate the declarative approaches in BPM, we will briefly describe some of their main exponents: the DECLARE language, the Dynamic Conditions Response (DCR) Graphs, and the Decision Modelling Notation. The following example illustrates a typical process description formulated through behavioral rules in a declarative way. It serves as the basis for the DECLARE map, the DCR Graph, and the DMN table in Figs. 1 to 3, respectively.

Example. A product ordering process specification consists of the following constraints:

- The process starts whenever a customer submits an order,
- Whenever a customer submits an order, it needs to be finalized by the seller. Conversely, a seller cannot finalize orders not submitted by the customer.
- Once an order is finalized, it must be confirmed or rejected.
- A confirmed order may be rejected later on (due to "late" problems), but not vice versa: a rejection cannot be reverted.
- An order can be confirmed only if it has been shipped before.
- An order may be rejected autonomously by the seller. However, if the warehouse notifies the seller of a shipment issue, then the seller has to reject the order.
- The warehouse decision is firm: either the order is shipped, or a shipment issue is notified.
- An order can be finalized at most once.

A valued customer is eligible for express delivery with expensive orders (based on the DMN table in Fig. 3).

DECLARE and LTL$_f$ Semantics. One of the declarative process modeling languages is DECLARE [19]. DECLARE specifications consist of a set of constraints over activities, each typically expressed using templates grounded in Linear Temporal Logic on Finite Traces (LTL$_f$). LTL$_f$ provides a formal semantics for specifying temporal properties of sequences of events, making it well-suited for reasoning about process executions. The repertoire of standard DECLARE templates aims to cover the most common patterns used in behavioral process verification (based on a work surveying those in the context of industrial applications [22]) while hiding the complexity of formal logics for their expression. For example, a RESPONSE(a,b) constraint between activities a and b dictates that whenever a occurs, b must eventually follow. In LTL$_f$, this can be expressed as $\Box(a \rightarrow \Diamond b)$, where $\Box$ denotes "globally" (always), and $\Diamond$ denotes "eventually". The templates are associated with a graphical

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

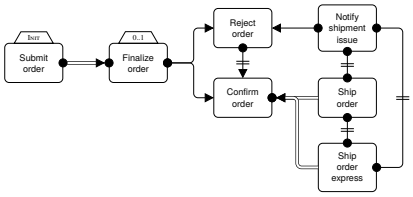


Fig. 1: A DECLARE map

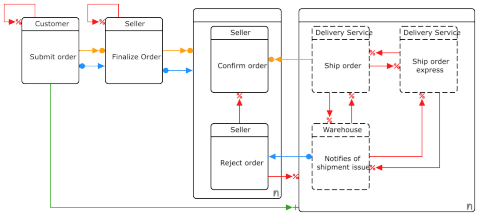


Fig. 2: A DCR Graph

notation. RESPONSE(a,b), e.g., is diagrammatically represented as 

.The example of an order handling process mapped in DECLARE is depicted in Fig. 1.

Dynamic Condition Response (DCR) Graphs. Another formalism in Declarative BPM is the Dynamic Condition Response (DCR) graphs [29]. While DCR graphs and DECLARE share some principles, i.e. activities are represented as nodes, and constraints as relations between nodes, other aspects separate the two notations: first, the DCR notation introduces a minimal set of constraints (four in its original presentation) compared to 18+ standard DECLARE patterns; second, its underlying semantics is given in terms of labeled transition systems; third, each activity node is labelled with an activity name and a finite set of roles, declaring which role can execute the activity; finally, DCR graphs support *defeasible* constraints, which cannot be directly mapped by DECLARE constraints. In short, the state of a DCR graph is given as a *marking* of each activity, recording if it has been executed in the past, whether it is pending, and whether it is currently excluded/irrelevant – depicted graphically as a dashed box (cf. Fig. 2). A relation from an activity a to another activity b either constrains the execution of the activity b (e.g., a condition relation from a to b declares that b can only be executed if 1) it is included, and 2) if a is included, then a must have been executed in the past) or defines a change of state of activity b, when activity a is executed. The notation has been extended with a range of additional features, including data, computations and guarded rules, time, subprocesses, and choreographies. In this paper, we focus on its classical definitions, using the nesting of activities to denote that a constraint is applied to each of its constituents.

Decision Model and Notation (DMN). While DECLARE and DCR graphs focus on temporal constraints, the Decision Model and Notation (DMN) complements the two notations by providing a standard for modeling decision logic. DMN separates decision-making from process flow, typically using decision tables to specify how outputs are derived from inputs (cf. Fig. 3). In a declarative setting, DMN can be used as a qualifier for activities, simplifying otherwise complex logic that would otherwise be directly embedded into process models. DMN has been integrated for both DECLARE and DCR graphs [47,30].

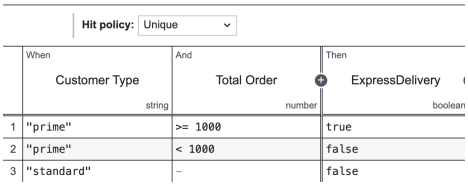


Fig. 3: A DMN table (excerpt)

3 Development of DBPM Curricula

The development of our educational practice insights takes as a departure point the experience of the author team teaching declarative and decision process models across

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

Table 1: Courses teaching DBPM material (a non-exhaustive sample).

Course	Eds.	Level	Audience	University	Country
Knowledge Management and Business Intelligence	16	MSc.	Business Engineering, Information Systems Engineering, Information Management, Computer Science	KU Leuven	BE
Business analysis	16	MSc.	Business Engineering, Information Systems Engineering, Information Management	KU Leuven	BE
Reactive and Event-Based System	6	BSc.	Computer Science	U. Copenhagen	DK
Development of Cloud-Based Health Apps	5	BSc.	Health Science, Global Health and Health Informatics	U. Copenhagen	DK
Process Mining	5	MSc.	Business Informatics, Data Science, Artificial Intelligence	U. Utrecht	NL
Health Data and Interoperability	4	BSc.	Public Health Science, Global Health and Health Informatics	U. Copenhagen	DK
IT and Business Process Modeling	3	BSc.	Business Informatics	IT University Copenhagen	DK
System Integration	3	MSc.	Computer Science	Technical University of Denmark	DK
Process Modeling	3	BSc.	Information Science, Computing Science	U. Utrecht	NL
Formal Foundations of Business Process Modeling	3	MSc.	Information Systems	WU Vienna	AT
Process Mining Applications of Data Science	2	BSc.	Business, Economics and Social Science	WU Vienna	AT
Process Oriented and Event-Based Systems	2	MSc.	Computer Science	Technical University of Denmark	DK
Process Mining	1	MSc.	Computer Science	Technical University of Denmark	DK

four countries in twelve courses across bachelor and master programs in Europe. Table 1 documents each of the courses, as well as their main intended audience (Bachelor or Master level), and their provenance (e.g., Information Systems or Computer Science). While the sample is not exhaustive, it aims at balancing geographical distribution and level of expertise, allowing us to compare different strategies when imparting DBPM material.

3.1 Stakeholders, their expected backgrounds, and coping with diversity

Overall, the courses where DBPM materials are addressed are in three different fields of education: computer science, business informatics, and health informatics. While some understanding of software development and IT sits at the intersection, the educational background and expected competencies are diverse. Students from business and healthcare informatics often have strong competencies in information systems, including performing fieldwork, organizational management, and enterprise architectures. This comes at the expense of foundational studies in computer science, and they have not been exposed to background material such as modal or temporal logics. On the other hand, the courses in computer science education require maths and experience in software development as prerequisites, but students have rarely worked on projects at the intersection with stakeholders (which is an ability required for process mapping/elicitation tasks). Between the two educations, the health informatics students have specialized knowledge in the healthcare domain and a few introductory courses in software engineering and databases.

The motivators for students in the three types of education differ. Feedback from health and business informatics students emphasizes *the value of modeling*, as in how models help stakeholders in the decision-making process (for instance, improving quality, automation, or optimization). In contrast, computer science students tend to appreciate the *technical challenges* associated with the declarative mindset. Often, computer science students

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

would engage in the programming of technical artifacts such as declarative process engines, verifiers, or extensions for process mining, making a transition from the use of foundational material to the creation of software artifacts.

Another differentiator sits at the core of the courses, where some specialize in the modeling and integration of models, while others focus on the implementation via process-aware information systems and process mining techniques. The inclusion of DBPM tools provides alternatives to students. Declarative process mining, for instance, may provide representations that differ from overly fitting procedural spaghetti models coming from direct-follows representations, and an easier description of business rules used for conformance checking vis-a-vis process alignments.

The teaching approach for DBPM material changes depending on the assumed background of the educational programs. For instance, in computer science education, teaching DECLARE based on its underlying LTL_f formulas aids understanding based on previous knowledge. For business and health informatics students, it may be counter-productive. In those cases, teaching using natural language patterns or positive/negative examples of satisfied templates is preferable.

In most of the considered courses, DBPM techniques cover a part of the overall content, alongside procedural approaches in BPM. The latter are typically supported by standardized and more mature modeling notations such as BPMN or workflow nets [21,1]. Thus, educators need to explain the position of declarative models as a tool part of the modeling repertoire. The emphasis given is mostly on explaining when declarative representations can be more succinct, easier to use, and explain to the business in terms of control flow, and how DMN can offload complex reasoning and fragmentation in procedural systems.

3.2 Integration of declarative models into the BPM lifecycle

In the next paragraphs, we will discuss the current role of declarative and decision models across the BPM lifecycle [21], as well as some opportunities for further support.

Identification. Typically, process scoping is one of the most difficult tasks for our students, especially for those who are more technically-minded. Not all courses consider scoping as a core activity in the curriculum, given the multiple types of learners' profiles. This phase, being concerned with high-level aspects such as the process landscape and value-chain analysis, does not usually define a specific process modeling notation.

Discovery. While the identification phase provides the list of candidate processes, this phase requires our students to lower the abstraction level by mapping their processes using multiple techniques. Depending on the course, this will involve ethnographies, interviews, document analysis, process mining, or user interface testing. In terms of curriculum development, this is the phase where students are introduced to the different process model notations. Declarative process notations are taught together with imperative modeling notations (e.g., BPMN, workflow nets, or UML activity diagrams).

Teaching the process discovery phase requires two main learning outcomes: first, to be able to elicit process information from potential stakeholders, and second, to familiarize students with the syntax and semantics of formal process modeling languages. These tend to be parallel skills, and given the short time spans, sometimes the courses compromise one of the aspects. For computer science students, this tends to be the process elicitation part. Instead, students are handed process descriptions for their modeling tasks. The nature of

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

descriptions may influence the resulting models. If the descriptions have been written with a specific notation in mind (e.g., a baseline BPMN model), their textual descriptions may be biased towards sequential flows [23]. Using the same description style to map declarative models will create *cognitive dissonance*, as students will need to map concepts foreign to the notations, such as flows and loops, resulting in over-constrained models. The second challenge here is the inherent ambiguity of natural language, which may contain unclear relations, underspecifications, unclear references, or inconsistencies in the text [42]. A final challenge comes from the *ontological misalignment* between imperative and declarative process notations. While essentially both imperative and declarative notions of processes describe traces, their metamodels clash with terms that are named differently but represent the same information (e.g., "effects" in some tools for DCR graphs correspond to "service tasks" in BPMN), or where the same term represents different information (e.g., BPMN *events* do not correspond to DCR *events*, as the latter ones encompass tasks in BPMN, too).

Analysis. Modeling is not a one-off phase, and students iteratively refine their models. They typically follow the process model quality assurance layers defined in [21], including syntactic quality (i.e., *does it conform to a valid model?* and *is it a sound model?*), semantic quality (i.e., *does it represent the real-world process?*), and pragmatic qualities (i.e., *does it conform to domain-specific policies?*). We will describe each of these steps below.

Some aspects in *syntactic quality* can be directly enforced by modeling editors for DCR [59,16], DECLARE [49,11], and DMN [37]. One aspect in this category involves the verification of models. While the students apply standard soundness definitions [21] ported to their declarative setting, particular extensions of declarative formalisms may render unsound models. For instance, declarative models may lead to processes with hidden constraints [15], inconsistencies [17], or that are time-locked [9]. Also, just as for imperative choreographies, one may model declarative choreographic models that result in distributed endpoints whose communications are unrealizable. This can be characterised formally for DCR Choreographies as *end-point projectability* [31]. Soundness itself may be difficult to identify with the naked eye, firstly because realistic declarative process models bear a high degree of interrelated constraints, and secondly because the interplay of multiple dimensions, such as time, data, and decisions may hide implicit dependencies. Currently, the concepts of soundness for declarative models are imparted in some of the courses, and the verification of soundness properties via model-checking techniques [55,39,40] is an active area of research, where students can learn the theory but face difficulties applying it, given the lack of tools supporting these aspects.

Multiple techniques contribute to the validation of semantic quality aspects of declarative process models. The most preferred option is simply a discrete-event simulation. This technique not only supports the model's validity but is also the most widely used for reinforcing understanding of the different types of constraints in a declarative model. Simulations also become the guardrails for subsequent model changes. Using test-driven development techniques [64], students define positive and negative control traces to verify whether changes to the constraint set affect model satisfiability. *Open tests* [12] are used whenever tests should be considered as partial specifications, where only the most important ordering relations between events are specified. Checking for process *completeness* is challenging, as it can only be addressed via validation with domain specialists, and does not pertain to specific problems originating from the use of declarative notations. For instance, initial models based on textual descriptions may be incomplete when students only map causal relations between activities, forgetting the produce-consume relations between artifacts generated by the multiple events. In addition, combinations of multiple

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

constraints might yield the same set of accepting traces, complicating comparisons with baseline solutions. As mitigation actions, students can make use of dual-coding tools to align process documentation, such as process descriptions, SOPs, and business rules, and their corresponding declarative process models using semantic annotations [43].

Pragmatic qualities relate to the usability of process models, and while multiple factors affect the comprehension of declarative process models [3,24,23], a salient aspect corresponds to the creation of modeling guidelines for declarative processes. For instance, an important characterization is that the declarative notations used consider events as first-class citizens, and while this notion is naturally refined into system, computational, and data-driven events [44], their ontological misalignment with imperative notions renders common guidelines for activity readability in imperative process models [46] unusable for declarative models. Another aspect concerns model layout: while imperative models have clearly defined starting points, their declarative counterparts may have multiple enabled events at the start of an execution. Tools like DCR-js [59] implement automatic layout techniques to reduce the number of salient constraints, minimize overlapping edges, and impose left-right alignments, but further research is needed to identify the most effective layout techniques for declarative models with full expressiveness.

Part of the analysis phase includes quantitative and qualitative analysis, and while some qualitative analysis techniques can easily be applied to a declarative mindset (e.g., value-added analysis), important analyses, such as waste analysis and cycle-time analysis, have not yet been described for DECLARE or DCR graphs. Some of the notions needed are either implicit (e.g., handovers) or not first-class primitives in existing declarative simulators, such as queues and resources on hold. Thus, teaching these aspects has so far been done using imperative specifications and simulation tools.

Redesign. The teaching of process redesign is normally focused on the methods rather than on the tools supporting them. At the same time, we are not aware of a process redesign tool specifically designed for declarative process models. Redesign takes place both as a natural step of refinement in the modeling phase, a predecessor to the implementation of processes, and the resulting step of value-based analysis. Notation change may be a useful opportunity in the redesign of processes. First, because strict process flows can be decomposed into precedence and obligatory constraints, adding flexibility in the orchestration of activities, but also because business adopters of the declarative mindset note flexibility, cost, and modeling time as some of the main advantages over traditional imperative approaches [36].

Two alternative workflows can support redesign initiatives. First, it is possible to depart from an imperative model and generate either a DECLARE or a DCR graph [8,62]. Such a model directly corresponds to the flows accepted in the BPMN model, but renders explicit the large number of constraints needed to do so. This is an opportunity to apply redesign patterns to offer more flexibility in the models. Direct encodings of BPMN models map to declarative models with numerous constraints, and students can change these dependencies to add more scenarios. The second possibility is to apply test-driven modeling techniques [63,57]: departing from an unconstrained model, generate representative traces, and gradually add constraints to understand their impact. While these strategies help outline the boundaries accepted by the process, a lack of functionality is the analysis of time and costs, which are covered in imperative models.

Implementation. Mostly catered to computer science students, this phase transitions from process models intended for communication to executable process models. This step requires extending models with additional dimensions and refined constraints, including

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

data and time, service activities (called *effects* in the tools for DCR graphs by DCR Solutions at DCRSolutions.net), and datatypes. This is also the phase where students need to define an architecture that enables them to interface with outside applications, for instance, by developing a presentation layer, defining the sources of events, and linking to a PAIS via existing APIs. A crucial prerequisite for this step is the involvement of industrial partners who implement APIs, thereby demonstrating their practicality in real-world settings. At the end of the course, students will have experimented and implemented both imperative and declarative models in exemplary PAISs, fostering critical thinking.

Process Mining/Monitoring. Material covered in declarative process mining lectures includes process discovery and conformance checking in terms of rule checking and alignments, for which specific process mining tools have been developed [28,13,33]. The selection of logs to evidence the applicability of declarative process mining techniques plays a key role here. If process variants are not properly distributed, the resulting models will be highly constrained and not benefit from a declarative model. Using models with high diversity (proper for the execution of knowledge-intensive processes [18]) may help to illustrate cases where declarative approaches benefit in the comprehension of the processes. Some examples of logs bearing these characteristics are the Sepsis and Traffic Fine Management logs [45,38]. Scalability of declarative process mining tools is also of consideration when students are building projects with their own logs. In these cases, tools like DCR4Py [28], Declare4Py [20], and MINERful [33] may provide an easy access link to their development workflow since they offer APIs and can be used as libraries. A clear teaching opportunity is the integration of PAIS implementations using imperative models and their monitoring using conformance checking techniques.

3.3 Instructional Approaches and Evaluation Techniques

This section describes the learnings from the evaluation techniques applied to the different courses, including Multiple Choice Questionnaires (MCQs), Project-Based Learning (PBL), flipped classrooms, and gamification. We proceed to discuss each of them below.

Multiple Choice Questionnaires. Being one of the most scalable alternatives, MCQs have been used to evaluate knowledge in DBPM when dealing with cohorts larger than 200 students. An exemplary exam question would require verifying whether LTL_f formulas and DECLARE constraints are satisfied by a set of traces, or complied with by workflow nets and finite-state automata. Our challenge lies in crafting questions that capture high-level ILOs as defined in the Bloom Taxonomy, as high-level objectives such as synthesis, evaluation, and analysis tend to be better mapped to other evaluation methods. Similarly, evaluating the skills in modeling declarative processes (an active skill) cannot easily be mapped to an MCQ test. In those cases, we resort to additional assignments during the course.

Project-Based Learning. PBL has been the preferred evaluation technique for mid-size courses (up to 100 students). In these cases, students implement an open-ended project engaging with an existing organization in a journey from process identification to recommendations or automation. The students are incentivized to apply modeling techniques from both imperative and declarative approaches, and reflect on the best notations to support their processes. Students assess how the different modeling techniques support the process, and this incentivizes them to explore imperative, declarative, and decision-based models, sometimes even complementary. Indeed, the resulting process landscapes may include models from multiple notations if the students deem it appropriate. This evaluation technique supports high-level intended learning outcomes such as modeling,

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

communicating, and implementing process-aware solutions, but it poses some challenges in the implementation of the coursework. First, it requires engagement with external companies, who sometimes lose interest over the course of the project. Second, it requires intuitive and robust tools for declarative tools at all levels of the lifecycle. Third, it benefits from a mixed student cohort of business informatics (who have experience in stakeholder management) and computer science (who have experience in the development of applications). Finally, these courses become difficult to scale given the amount of follow-up required for projects. An alternative to reduce the risks associated with external partners used is to provide closed projects. For instance, one of the courses resorts to LTL$_f$ and DECLARE for the specification of a process-aware system in the education domain itself, encompassing the processes regarding enrollment, the organization and attendance of courses, and the release of diplomas. LTL$_f$ and declarative constraints are used to specify the inter- and intra-dependencies of those processes, focusing on the control-flow dimension.

Flipped Classrooms. In flipped classrooms, the instructional approach reverses the traditional learning model by shifting direct instruction from the classroom to the individual learning space. We have explored flipped classroom education for DPMs, where videos and lecture material are provided beforehand. The challenge with this approach is the lack of accessible self-study material developed for Declarative models. Except for the 2022 Process Mining Summer School's lecture notes [19], most online material resorts to research papers, PhD theses, or company manuals for specific features, limiting the possibilities for students to acquire knowledge independently. Moreover, concepts explored in research papers may be modified once the declarative language increases in expressiveness (e.g., do the same criteria for soundness apply when we move to declarative models with timing constraints?). While advantageous in theory, student evaluations indicate that students still prefer the clarity of a lecture explanation followed by exercise sessions over the current implementation of flipped classroom education.

Gamification. To counter the heavily identified cognitive load that declarative models pose on modelers and users of DBPM, gamification offers a way to bring relief in the form of visual or salient features that render the modeling process into a more tangible or playful environment. [14] introduced extra visualization allowing users to create extra graphs and annotations explaining dependencies in DECLARE models while executing and simulating behavior, fostering an inquisitive approach to understanding the models. [35,61] introduced virtual 3D environments to visualize the context and real-life execution of processes modeled in DCR Graphs, providing an immersive experience that helps users grasp intricate relations but also fosters a more natural relation with the modeling context.

3.4 Overall student perception

Typically, high-level process mappings tend to show sequential operations that are better mapped to imperative notations such as BPMN. However, when students work on projects involving a degree of reactive behavior (e.g., mapping user interfaces, games, and healthcare processes) or explicit constraints (e.g., legal or safety regulations), they begin to appreciate the flexibility and explicit modelling of rules offered by DBPM notations.

The DBPM-related topics within the broader courses in which they are taught spark interest in the learners, who often select those as the themes for their BSc. or MSc. theses and research topics. To briefly give an account of it, we take as a sample the academic year 2025-2026 in a Dutch University where Process Mining and Process Modeling are taught. At the time of writing, one Bachelor's and two Master's honors paths (research projects

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

held by students with outstanding curricula), and five Master's and one Bachelor's theses are specifically centered around declarative process mining and modeling. The students who selected these topics tend to have a strong technical or formal background. The novelty of the approach also calls for innovation projects, and some of our students have created startups based on declarative modelling techniques (integrated with imperative models).[5] This aspect can be seen as a sign of the relative novelty of the approach, more at the stage of ongoing research for computing scientists in BPM than supported by mature solutions.

4 Challenges in the education of DBPM

Based on the experience gained by teaching declarative models in the different lifecycle phases documented in Sect. 3.2, we summarize some of the most pressing challenges in the pedagogical integration of declarative models in three overarching themes: development of pedagogical material, challenges in the understandability of DBPM, and tool support. We will describe them in further detail below.

Development of Pedagogical Materials. One of the greatest barriers to the adoption of declarative techniques is the absence of pedagogical materials for bachelor's or master's students. Classical teaching materials used by the BPM community [21] were designed with BPMN in mind, and a direct application of exercises is likely to yield counterproductive outcomes. For instance, baseline process description repositories (e.g., [10]) are written in a style that best fits an imperative mindset, sometimes even with references to BPMN elements such as XOR gateways and exception signals. Teachers must create their own syllabus, examples, and exams using a combination of sources from research articles, theses, and manuals. While some books and MOOCs describing declarative models exist (e.g., [53]), they do not consider the impact of DBPM throughout the process lifecycle. This, compared with BPMN or other techniques, becomes a deterrent to adopters and a barrier to self-study. Moreover, it adds risks to the sustainability of the courses when teachers change or become ill. We believe that a common effort should be made to describe the foundations of the declarative mindset. This effort, including foundations, examples, teaching material, and evaluations, would benefit from a notation-agnostic approach, so modelers can relate to the foundations rather than the specifics of DECLARE or DCR. Moreover, teaching material should be customizable for different types of students interested in the area, so both business informatics and computer science students can benefit from it.

While we advocate for a pedagogical approach that integrates multiple modeling styles, we must consider the possible carryover effects coming from the teaching of different notations in sequence. For instance, if students become familiar with concepts such as process flows using an imperative notation, they may develop mental models that influence the quality of declarative models developed afterwards: the latter may be overconstrained to accept only actions in a sequences. An alternative is to reverse the order, introducing declarative approaches first, and then the imperative ones. We hypothesize that, in this way, students may first learn how to think "declaratively", and later simplify notions by taking into account only specific action unfoldings.

A second challenge corresponds to the identification of language features and associated pedagogical content. Both DECLARE and DCR have been extended since their original

[5] https://ahead.is/

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

presentation to model scenarios where more than control flow is defined. These constructs (data, time, organizational resources, access control, collaborations, computations, choreographies, subprocesses, etc.) add complexity in understanding, modeling, and maintaining models. Moreover, the selection of cases for modeling is just as important. If cases come from closed and controllable environments, students may not appreciate the added complexity of declarative approaches. An alternative would be to capitalize on the interaction with companies and students applying project work in an organization to identify possible cases. For instance, Fig. 4 shows the notation recommendation after students performed process mapping and modeling of real organizational processes (including fieldwork, interviews, and validation). Here, it is important to note an almost even split between BPMN and DCR. These domains may also help students identify similar companies to work in projects with domains that benefit more from a declarative mindset.

The third challenge is the adaptation to the varying sizes of BPM courses. Our courses in Tab. 1 have cohort sizes ranging from 20 to 500+ students per edition. Thus, the materials and evaluation strategies need to be adapted accordingly. Student evaluations gravitate towards a preference to work on projects rather than on theory assignments, but project-based work with companies faces scalability and coordination issues. Currently, our courses are considering hybrid approaches, where instructors define the projects. While this type of coursework faces ecological validity constraints, it is logistically easier to implement. Hence, it would be beneficial to create project banks of Knowledge-intensive workflows, especially designed for ILOs related to modeling declarative specifications.

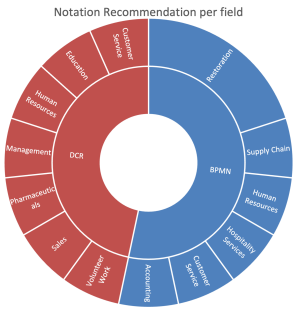


Fig. 4: Notation selection after field work and process mapping in BPMN and DCR graphs. IT and Business Process Modelling course, 2020.

Tools. A challenge is the limited access to robust, user-friendly tools for novice students in DBPM. Although modeling is central across the BPM lifecycle, most focus remains on discovery. Existing editors for DECLARE [5,11,51,49] and DCR graphs [59] do not always support simulation or analysis. Some are also difficult to deploy at scale; for instance, requiring Java installations for large cohorts (500+ students) is impractical. In contrast, many of the imperative modeling tools we have used in our courses are more mature and cover all lifecycle stages, creating an uneven comparison driven by tool support. DMN tools, however, are better integrated into platforms like Camunda and Signavio, which simplifies adoption, documentation, and support. Notably, the industrial[6] offering of DCR graphs supports simulation and as for the academic [13] tools need no installation and are free for academic use, facilitating an initial adoption of the technologies by novice users.

Tool design must also reflect user profiles. Students are novice users whose early experiences shape adoption. Moreover, business informatics and computer science students have different needs: the former focus more on modeling and analysis, while the latter emphasize software development. Thus, graphical editors and analysis tools are more relevant for practice-oriented students, declarative process APIs are essential for prototyping, and execution APIs and PAIS with support for declarative models are

[6] DCRSolutions.net

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

essential for process execution. An open REST API has been available for DCR[7] since 2017. Encouragingly, recent work has also introduced new APIs for DECLARE and DCR [20,28], and there is support for execution environments for pure declarative [58,33,7].

Understandability. While there is a body of work investigating the understandability of declarative models [23,3,24], we believe that understandability goes beyond the sensemaking abilities against a process model, and it includes understanding the underlying process that the model is representing. In the context of teaching, this reflects in integrating different representations of the process as imperative and declarative models. In addition, this also refers to the originating documentation of the process, where (imperative) process descriptions may be combined with compliance rules.

A separate part refers to the understandability of the notations. Being originally developed by formal methods practitioners, the symbols used to denote constraints may not be understandable by novice users. Semantic opacity [48] describes when a novice cannot guess what the symbol means by looking at it. The geometric shapes used in the notations require memorization because there is no logical link between the visual representation and the underlying concept. We see recent work in the redefinition of the DCR and DECLARE notations based on the concepts of the Physics of Notations [11,60], but more language features considered and more empirical evidence of their effectiveness are still required.

5 Conclusion with an agenda for DBPM in the classroom

With this paper, we account for the potential of DBPM in an educational context by showcasing multiple instances where declarative methods are being used in the classroom. As a compendium to our manuscript, we make available a (growing) sample of the DBPM teaching material developed during the years at the following address: github.com/Process-in-Chains/Teaching-Declarative-BPM.
Next, we conclude with a summary of our experience and an agenda for future work that is based thereupon, followed by a discussion of the limitations of our present contribution.

5.1 An agenda for DBPM in the classroom

The experience discussed throughout this paper reveals many common threads in DBPM. Drawing on these observations, we outline an agenda for closing the gap between the state of the art in declarative process modeling and its uptake in education.

Systematic modeling approaches and scoped curricula. A recurring difficulty in teaching declarative BPM is the absence of a structured build-up in model notation and subsequent use: students are often exposed to the full constraint vocabulary at once, without guidance on which constructs matter most, for whom, and in what order. Nagel and Delfmann [50] take a step in this direction with their structured collection of DECLARE tasks of increasing difficulty, ranging from basic concepts and single constraints through to combinations, model behavior, and inconsistency. Future work could develop analogous structured curricula for DCR, and establish a principled account of which constructs are relevant to which audiences. Given most students will have curricula based on seminal BPM works following the BPM lifecycle [21], these can be used as stepping stones for introducing where declarative approaches excel and how certain complexities might not be relevant for

[7] documentation.dcr.design/documentation/using-postman-to-access-dcr-active-repository/

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

all stages. For instance, modeling might rely on simple constraints, while more intricate analyses might convert to the full expressive power of declarative models.

Closing the gap with imperative approaches. A persistent barrier to adopting declarative notations in education is that students typically arrive having learned BPMN or similar imperative languages first [54,23]. Rather than treating imperative and declarative modeling as alternative paradigms to be taught in isolation, a more productive strategy is to teach them in dialogue. Conversion tools that bridge the two paradigms offer one lever here: recent work on compositional mappings from BPMN to DCR Graphs [62] and from sound workflow nets to LTL_f specifications [8] gives educators concrete machinery for asking students to translate between representations, thereby exposing the structural differences in what each paradigm makes explicit or leaves implicit. Jalali [34] showed that perceived ease of use and usefulness of DCR and CMMN varied across student cohorts trained over several weeks, and that perceptions could shift meaningfully with feedback, suggesting that structured cross-paradigm exposure rather than isolated training in one notation may produce a more robust understanding. The empirical work of Haisjackl et al. [27] provides one of the most consistent findings, which is that prior experience with imperative notations can actively impede understanding of declarative models, particularly where graphical representations look superficially similar but carry different semantics.

Test-driven modeling. Test-driven development has long been used in DBPM education. Zugal et al. [63,64] showed that test cases reduce cognitive load and improve the quality of DECLARE specifications, while Slaats et al. [57] extend this methodology with a formal theory of iterative, test-driven model development for DCR. Test-driven methodologies have, however, not yet been studied systematically as a *pedagogical* method: the question of whether introducing test cases early in a modeling course improves learning outcomes, or helps develop an intuition for declarative semantics, remains open. Similarly, the automaton-based comparison approach in [56], which makes the semantics of DECLARE specifications visible as finite automata, could be used as a teaching aid, giving students a formal and grounded account of what their models actually permit and prohibit.

AI to the rescue? The broader educational technology landscape has seen rapid development of AI-powered tutoring and feedback systems. The recent DUET prototype [26] demonstrated LLM-based automated feedback on student-submitted UML and ER diagrams, finding that the system could surface structural differences between reference and student models and generate reflective, targeted feedback at scale. EduClare [51] already takes steps in this direction for DECLARE, combining intelligent tutoring principles with large language models. The growing literature on generative AI for conceptual modeling more broadly [25], spanning UML, ER, and BPMN, provides further methodological grounding for extending this line of work to declarative notations. Declo [4], which allows users to specify DECLARE constraints in natural language without prior knowledge of the notation, further reduces the syntactic barrier. The elicitation of DCR models using LLMs has been explored in [41], identifying challenges for LLMs in constraint identification.

Lessons from adjacent modeling areas. The broader process and conceptual modeling education literature offers transferable lessons. Works on teaching BPMN, including exercise formats, error taxonomies, and real-world case studies as modeling targets, represent a body of pedagogical knowledge that can inform how DECLARE and DCR courses are designed [52]. The gamification approaches point toward engagement-oriented design. Still, some of these initiatives remain unevaluated for actual learning outcomes.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

5.2 Limitations

This paper covered various teaching methodologies and declarative languages. However, we covered only the three most prevalent approaches that typically appear in research. Other declarative-minded BPM approaches such as Proclets [2], Guard-Stage Milestone lifecycles [32], or Object-Centric Behavioral Constraints [6] are not covered. Furthermore, we consider a limited set of courses taught in six institutions, primarily situated in computer science and information systems education. The viewpoint also stems from the teacher's perspective with relatively limited student input. Finally, we mostly base our discussion on experience with academic education, without highlighting the professional development that some languages enjoy through tool vendors' tutorials and educational support.

Acknowledgments. The authors want to thank Anti Alman, Paul Cosma, Søren Debois, Patrick Delfmann, Rik Eshuis, Marco Montali, Kate Revoredo, Tijs Slaats, Jan Vanthienen, and Jan Martijn van der Werf for their precious support over the years co-teaching DBPM.

References

1. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1), 21–66 (1998)
2. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A framework for lightweight interacting workflow processes. Int. J. Cooperative Inf. Syst. **10**(4), 443–481 (2001)
3. Abbad Andaloussi, A., Davis, C.J., Burattin, A., López, H.A., Slaats, T., Weber, B.: Understanding quality in declarative process modeling through the mental models of experts. In: BPM. pp. 417–434. Springer (2020)
4. Alman, A., Balder, K.J., Maggi, F.M., van der Aa, H.: Declo: A chatbot for user-friendly specification of declarative process models. In: BPM Workshops. vol. 2673, pp. 122–126. CEUR (2020)
5. Alman, A., Di Ciccio, C., Maggi, F.M., Montali, M., van der Aa, H.: RuM: Declarative process mining, distilled. In: BPM. pp. 23–29. Springer (2021)
6. Artale, A., Calvanese, D., Montali, M., van der Aalst, W.M.P.: Enriching data models with behavioral constraints. In: Ontology Makes Sense. pp. 257–277. IOS Press (2019)
7. Awad, A., Awaysheh, F., López, H.A.: Best: a unified business process enactment via streams and tables for service computing. Process Science **2**(1), 23 (2025)
8. Barbaro, L., Varricchione, G., Montali, M., Di Ciccio, C.: From sound workflow nets to LTLf declarative specifications by casting three spells. In: BPM Forum. pp. 3–22. Springer (2025)
9. Basin, D., Debois, S., Hildebrandt, T.T.: In the nick of time: Proactive prevention of obligation violations. In: CSF. pp. 120–134. IEEE (2016)
10. Bellan, P., van der Aa, H., Dragoni, M., Ghidini, C., Ponzetto, S.P.: Pet: an annotated dataset for process extraction from natural language text tasks. In: BPM. pp. 315–321. Springer (2022)
11. Blasilli, G., Ferro, L.S., Lenti, S., Maggi, F.M., Marrella, A., Catarci, T.: Improving the understandability of declarative process discovery results using easydeclare. Information Systems p. 102667 (2025)
12. Christfort, A.K., Cosma, V.P., Debois, S., Hildebrandt, T.T., Slaats, T.: Static and dynamic techniques for iterative test-driven modelling of dynamic condition response graphs. Data & Knowledge Engineering **157**, 102413 (2025)
13. Christfort, A.K., López, H.A.: DCR-JS: An online environment for declarative process mining. In: BPM Demos. pp. 256–263. CEUR-ws (2025)
14. De Smedt, J., De Weerdt, J., Serral, E., Vanthienen, J.: Gamification of declarative process models for learning and model verification. In: BPM. pp. 432–443. Springer (2016)
15. De Smedt, J., De Weerdt, J., Serral, E., Vanthienen, J.: Discovering hidden dependencies in constraint-based declarative process models for improving understandability. Information Systems **74**, 40–52 (2018)

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

16. Debois, S., Hildebrandt, T.T., Marquard, M., Slaats, T.: The DCR graphs process portal. In: BPM (Demos). pp. 7–11 (2016)
17. Di Ciccio, C., Maggi, F.M., Montali, M., Mendling, J.: Resolving inconsistencies and redundancies in declarative process models. Information Systems **64**, 425–446 (2017)
18. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive Processes: Characteristics, requirements and analysis of contemporary approaches. J. Data Semantics **4**(1), 29–57 (2015)
19. Di Ciccio, C., Montali, M.: Declarative process specifications: reasoning, discovery, monitoring. In: Process mining handbook, pp. 108–152. Springer (2022)
20. Donadello, I., Riva, F., Maggi, F.M., Shikhizada, A.: Declare4Py: A Python library for declarative process mining. In: BPM (PhD/Demos). pp. 117–121. CEUR (2022)
21. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of business process management. Springer (2018)
22. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE. pp. 411–420 (1999)
23. 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: BPMDS 2009. pp. 353–366. Springer (2009)
24. Figl, K., Di Ciccio, C., Reijers, H.A.: Do declarative process models help to reduce cognitive biases related to business rules? In: ER. pp. 119–133. LNCS, Springer (2020)
25. Fill, H.G., Horkoff, J., Fettke, P., Köpke, J.: Generative ai and conceptual modeling: H.-g. fill et al. Business & Information Systems Engineering pp. 1–5 (2026)
26. Gürtl, S., Schimetta, G., Kerschbaumer, D., Liut, M., Steinmaurer, A.: Automated feedback on student-generated UML and ER diagrams using large language models. arXiv preprint arXiv:2507.23470 (2025)
27. Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding declare models: strategies, pitfalls, empirical results. SoSyM **15**(2), 325–352 (2016)
28. Hermansen, S.V., Jónsson, R., Kjeldsen, J.L., Slaats, T., Cosma, V.P., López, H.A.: DCR4Py: A PM4Py library extension for declarative process mining in Python. In: ICPM Demos. CEUR (2024)
29. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. arXiv preprint arXiv:1110.4161 (2011)
30. Hildebrandt, T.T., Normann, H., Marquard, M., Debois, S., Slaats, T.: Decision modelling in timed dynamic condition response graphs with data. In: BPM. pp. 362–374. Springer (2021)
31. Hildebrandt, T.T., Slaats, T., López, H.A., Debois, S., Carbone, M.: Declarative choreographies and liveness. In: FORTE. pp. 129–147. Springer (2019)
32. Hull, R., Damaggio, E., Masellis, R.D., Fournier, F., Gupta, M., III, F.F.T.H., Hobson, S., Linehan, M.H., Maradugu, S., Nigam, A., Sukaviriya, P.N., Vaculín, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: DEBS. pp. 51–62. ACM (2011)
33. Iacometta, C., Di Ciccio, C.: Declarative process mining with MINERful, reloaded. In: BPM Demos & Resources. pp. 328–335 (2025)
34. Jalali, A.: Evaluating user acceptance of knowledge-intensive business process modeling languages. SoSyM **22**(6), 1803–1826 (2023)
35. Jensen, S.J., López, H.A.: Towards immersive environments for declarative process models. In: BPM. pp. 214–231. Springer (2024)
36. Keramidis, P., Hildebrandt, T.T.: On the business value of declarative business process management. In: HICSS. pp. 5631–5640 (2026)
37. Laurson, Ü., Maggi, F.M.: A tool for the analysis of dmn decision tables. In: BPM 2016 Demonstration Track. pp. 56–60. CEUR-ws (2016)
38. de Leoni, M.M., Mannhardt, F.: Road traffic fine management process (2015). https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

39. de Leoni, M., Felli, P., Montali, M.: Integrating bpmn and dmn: modeling and analysis. Journal on Data Semantics **10**(1), 165–188 (2021)
40. Lindner, J.: Exploring the Model Checking Problem of DCR Graphs. Master's thesis, Technical University of Denmark (Jan 2026)
41. Lindner, J., López, H.A.: Discovering declarative processes in textual descriptions using large language models. In: ICPM Workshops (2025)
42. López, H.A., Feng, B., Lindner, J., Franceschetti, M., Abbad-Andaloussi, A.: Ambiguity detection in business process descriptions: An evidence and an automated approach. In: BPM. pp. 379–396. Springer (2025)
43. López, H.A., Hildebrandt, T., Debois, S., Marquard, M.: The process highlighter: From texts to declarative processes and back. In: BPM Demos. pp. 66–70. CEUR-ws (2018)
44. López, H.A., Simon, V.D.: How to (re) design declarative process notations? A view from the lens of cognitive effectiveness frameworks. In: POEM. CEUR-ws (2022)
45. Mannhardt, F.: Sepsis cases - event log (2016). https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
46. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Information Systems **35**(4), 467–482 (2010)
47. Mertens, S., Gailly, F., Poels, G.: Enhancing declarative process models with dmn decision logic. In: BPMDS. pp. 151–165. Springer (2015)
48. Moody, D.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on software engineering **35**(6), 756–779 (2009)
49. Nagel, S., Amann, E., Delfmann, P.: declare-js: A web-based viewer and editor for declarative process models. In: ICPM Demos. CEUR (2023)
50. Nagel, S., Delfmann, P.: Towards an e-learning approach for declarative process modeling. In: BPM Forum. pp. 354–364. Springer (2024)
51. Nagel, S., Wolters, A., Riehle, D.M., Delfmann, P.: EduClare - An intelligent tutoring chatbot for teaching declarative process modeling. In: ICPM Doctoral Consortium/Demo (2024)
52. Recker, J., Rosemann, M.: Teaching business process modelling: experiences and recommendations. CAIS **25**(1), 32 (2009)
53. Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies, vol. 54. Springer (2012)
54. Reijers, H.A., Slaats, T., Stahl, C.: Declarative modeling–an academic dream or the future for BPM? In: BPM. pp. 307–322. Springer (2013)
55. Roveri, M., Di Ciccio, C., Di Francescomarino, C., Ghidini, C.: Computing unsatisfiable cores for LTLf specifications. JAIR **80**, 517–558 (2024)
56. Schützenmeier, N., Käppel, M., Ackermann, L., Jablonski, S., Petter, S.: Automaton-based comparison of declare process models. SoSyM **22**(2), 667–685 (2023)
57. Slaats, T., Debois, S., Hildebrandt, T.: Open to change: A theory for iterative test-driven modelling. In: BPM. pp. 31–47. Springer (2018)
58. Slaats, T., Mukkamala, R.R., Hildebrandt, T., Marquard, M.: Exformatics declarative case management workflows as DCR graphs. In: BPM. pp. 339–354. Springer (2013)
59. Tamo, L.K., Abbad-Andaloussi, A., Trinh, D.M.T., López, H.A.: An open-source modeling editor for declarative process models. In: CoopIS. vol. 3552, pp. 1–5. CEUR-ws (2023)
60. Trinh, D.M.T., Abbad-Andaloussi, A., López, H.A.: On the semantic transparency of declarative process models: the case of constraints. In: CoopIS. pp. 217–236. Springer (2023)
61. Villalobos, J.C., Jensen, S.J., López, H.A.: 3dcr: A tool for immersive process mining. In: ICPM Demos. CEUR-ws (2024)
62. Zhou, Y., López, H.A.: A compositional approach to mapping bpmn to dcr for cross-paradigm process modeling. In: FormaliSE. IEEE (2026)
63. Zugal, S., Pinggera, J., Weber, B.: Creating declarative process models using test driven modeling suite. In: CAISE. pp. 16–32. Springer (2011)
64. Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. In: BPMDS. pp. 163–177. Springer (2011)

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-36710-5_21

This document is a pre-print copy of the manuscript
(López et al. 2026)
published by Springer (available at link.springer.com).

The final version of the paper is identified by DOI: 10.1007/978-3-032-36710-5_21

References

López, Hugo A., Claudio Di Ciccio, Thomas Hildebrandt, and Johannes De Smedt (2026). "Teaching Declarative BPM: Where, When and to Whom it Matters". In: *BPM Educators Forum*. Ed. by Krzysztof Kluza, Karolin Winter, Lucineia Heloisa Thom, Amy Van Looy, Marco Comuzzi, Stephan Fahrenkrog-Petersen, Manfred Reichert, Joerg Evermann, and Barbara Re. Vol. 16936. Lecture Notes in Business Information Processing. Springer, pp. 328–347. ISBN: 978-3-032-36710-5. DOI: 10.1007/978-3-032-36710-5_21.

BibTeX

```
@InProceedings{   Lopez.etal/BPMEdu2026:TeachingDeclarativeBPM,
  author        = {L{\'o}pez, Hugo A. and Di Ciccio, Claudio and Hildebrandt,
                   Thomas and De Smedt, Johannes},
  booktitle     = {BPM Educators Forum},
  title         = {Teaching Declarative BPM: Where, When and to Whom it
                   Matters},
  year          = {2026},
  pages         = {328--347},
  crossref      = {BPM2026Edu},
  doi           = {10.1007/978-3-032-36710-5_21},
  keywords      = {Declarative BPM; Declare; DCR graphs; DMN; Education in
                   BPM}
}
@Proceedings{     BPM2026Edu,
  title         = {Business Process Management: Responsible {BPM} Forum,
                   Process Technology Forum, Educators Forum - {BPM} 2026
                   RBPM, PT, and Educators Forum, Toronto, ON, Canada,
                   September 27-October 2, 2026, Proceedings},
  year          = {2026},
  editor        = {Kluza, Krzysztof and Winter, Karolin and Thom, Lucineia
                   Heloisa and Van Looy, Amy and Comuzzi, Marco and
                   Fahrenkrog-Petersen, Stephan and Reichert, Manfred and
                   Evermann, Joerg and Re, Barbara},
  isbn          = {978-3-032-36710-5},
  publisher     = {Springer},
  series        = {Lecture Notes in Business Information Processing},
  volume        = {16936}
}
```