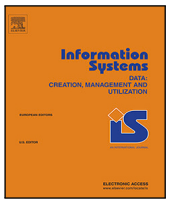
Information Systems 138 (2026) 102667

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Improving the understandability of declarative process discovery results using easyDeclare☆

Graziano Blasilli [a], Lauren S. Ferro [b], Simone Lenti [a], Fabrizio Maria Maggi [c], Andrea Marrella [a,*], Tiziana Catarci [a]

[a] Sapienza University of Rome, via Ariosto 25, 00185 Rome, Italy
[b] RMIT University, Melbourne, Victoria, Australia
[c] Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy

ARTICLE INFO

Keywords:
Understandability of declarative process discovery results
Process modeling languages
Declare
easyDeclare visual notation
Design principles

ABSTRACT

Declarative process models allow us to capture the behavior of a business process through temporal constraints on the evolution of process activities. In process mining, declarative process discovery focuses on deriving these constraints from event logs. Although the semantic aspects of declarative processes have been extensively investigated, there has been less focus on designing declarative visual notations that enhance model understanding and support analysts in solving process mining tasks. To improve the human understandability of declarative process models, in this paper, we present easyDeclare, a novel visual notation to specify declarative process models using the Declare language. easyDeclare was developed with consideration of the well-established Moody's design principles. We conducted extensive user experiments to demonstrate that easyDeclare, when compared with the original graphical representation of Declare, reduces the cognitive load required to interpret Declare models of increasing complexity, making it a promising alternative to enhancing overall comprehension of declarative process discovery tasks.

1. Introduction

One of the most common ways to visualize and communicate process mining insights is to use process models that show the actual flow of activities, events, and resources involved in a business process as recorded in an event log. Process models can be represented through many languages (e.g., BPMN, Directly-Follow Graphs, Petri-Nets, etc.) that visually map the overall process structure and support process analysts in identifying problematic areas (e.g., congestion, bottlenecks) and improving decision-making. The processes (and their issues) that such languages map out ultimately need to be read by business stakeholders, who may lack the required expertise in process analytics, but have to make decisions based on their results. Therefore, delivering high-quality and clear process model visualizations is an essential prerequisite for accurately conveying the behavioral knowledge of a process and guiding the selection of the most suitable visual analytics techniques, which help make sense of the multifaceted information hidden within the event log that record observed process executions [1]. Indeed, while process models provide a kind of "bidimensional" knowledge, visual analytics techniques extend this by integrating a broader range of interconnected insights, enabling interactive exploration of factual evidence derived from the event log [2]. However, the issue with using such process-oriented languages is their level of interpretability for persons who are not inherently from the field of computer science.

Despite the abundance of languages available for process models, it is widely acknowledged that most process modeling languages fall somewhere on the imperative-declarative spectrum. Imperative notations such as BPMN facilitate the description of sequential process flows, whereas declarative specifications like Declare describe cause–effect temporal relations between events [3]. Although the semantic aspects of declarative processes have been extensively researched, there has been comparatively less focus on designing declarative visual notations that enhance model understanding and incorporating features into these notations that facilitate comprehension as model complexity grows [4].

State-of-the-art solutions, e.g., [3,5,6], struggle with effectively communicating explicit concepts of how to interpret a declarative process model. Existing literature suggests that a new notation easing

☆ This article is part of a Special issue entitled: 'Process M. & Visual A.' published in Information Systems.

* Corresponding author.
E-mail addresses: blasilli@diag.uniroma1.it (G. Blasilli), lauren.ferro@rmit.edu.au (L.S. Ferro), lenti@diag.uniroma1.it (S. Lenti), maggi@inf.unibz.it (F.M. Maggi), marrella@diag.uniroma1.it (A. Marrella), catarci@diag.uniroma1.it (T. Catarci).

https://doi.org/10.1016/j.is.2025.102667
Received 5 August 2024; Received in revised form 17 October 2025; Accepted 12 December 2025
Available online 22 December 2025

understandability is needed [4,7]. This need arises because current notations can become overwhelming [8] and confusing [9,10], especially for non-expert stakeholders [11] who need to interpret these models to make informed business decisions. Given the importance of declarative models in accurately reflecting the flexible nature of business processes, improving their interpretability is critical for leveraging the full potential of process mining.

Drawing from the early development of the Verto modeling language [12], the notation presented in this paper, called easyDeclare, is designed to ease the process of understanding declarative process models. The development of this language has been done with consideration of the well-established Moody's design principles (PoN) [13] and several parameters (e.g., use of shapes) to maintain a contextual fit. easyDeclare aims to bridge the gap between the complexity of declarative models and the practical need for clarity and interpretability by business stakeholders. It incorporates intuitive visual elements that simplify the mapping of complex relationships and constraints within process models, thereby reducing cognitive load and enhancing overall comprehension.

To assess how well the proposed notation aligns with the PoN principles, we conducted expert interviews with scholars in the field. Through this process, we also evaluated the degree of alignment of Declare with each of the nine PoN dimensions. We finally developed a software tool to support the creation of Declare models based on our proposed graphical notation.

We conducted extensive user experiments whose results highlight that easyDeclare demonstrates superior effectiveness and efficiency in complex tasks and higher user satisfaction in ease of use and intention to use if compared with the original graphical representation of Declare. This makes it a promising tool for improving the communication of process mining insights related to declarative process discovery tasks to a broader audience, ultimately supporting better decision-making and process optimization.

The rest of the paper is organized as follows. Section 2 first introduces the required background on declarative processes and the development of visual notations necessary to understand the paper. Then, it discusses some state-of-the-art efforts aimed to improve the interpretability of Declare models by means of alternative graphical notations. Section 3 reports on an expert evaluation of Declare's understandability and details how its findings guided the design of easyDeclare. Section 4 assesses the alignment of Declare and easyDeclare with the PoN principles. Section 5 presents the results of the comparative evaluation conducted to assess easyDeclare against the original graphical representation of Declare, focusing on performance and perception metrics. Section 6 details the realization of EDD, a software tool for defining Declare models through easyDeclare, showing its usability through a dedicated user experiment. Finally, Section 7 concludes the paper.

The easyDeclare notation and the EDD tool are available under the *easyDeclare* GitHub organization at https://github.com/easyDeclare.

2. Background and state of the art

In this section, we introduce basic background concepts necessary for comprehending the content of the paper.

2.1. Declare

Declare is a declarative process modeling language introduced by Pesic and van der Aalst in [3]. Declare is qualified as "declarative" because it does not explicitly specify every possible sequence of activities leading from the start to the end of a process execution. Instead, it bases models on a set of constraints that must hold true during the execution of the process. All behaviors that respect those constraints are allowed. Constraints are applied to sets of activities and mainly pertain to their temporal ordering. In particular, Declare specifies an extensible set of standard templates (see Table 1) that a process analyst can use to model a process. Constraints are concrete instantiations of these templates. Declare is equipped with a formal semantics on Linear Temporal Logic on Finite Traces (LTL_f) [14], but the use of templates makes model comprehension independent of the logic-based formalization. Indeed, analysts can work with the graphical representation of templates while the underlying formulas remain hidden. Graphically, a Declare model is a diagram in which activities are represented as nodes (labeled rectangles) and constraints as arcs between activities.

Compared with procedural approaches, Declare models are more suitable for describing processes operating in unstable environments and characterized by numerous exceptional behaviors. Since anything not explicitly specified is allowed, a few constraints can specify many possible behaviors at once. Declare templates can be divided into four main groups: *existence templates*, *relation templates*, *mutual relation templates*, and *negative relation templates*. The first group consists of unary templates, which can be expressed as predicates over a single parameter. The remaining groups correspond to binary predicates over two parameters. The Declare templates with their semantics and graphical notations are summarized in Table 1.

Example 2.1 (*A Flight Booking Process Model*)**.** The scenario presented here illustrates the process a flight passenger follows from booking a ticket to boarding the aircraft. The corresponding declarative model is depicted in Fig. 1. The process begins with booking the ticket, represented in the model by activity Start booking. Passengers' personal information can then be provided and enriched with payment details. This submission is denoted as activity Provide data in the figure. The subsequent step is the authorization of the ticket payment (Pay), which triggers the completion of the booking phase (Complete booking). The authorization is eventually followed by the actual transfer of money (Complete transaction). As long as Check-in for the flight has not occurred, customers can still modify the provided data, such as changing the date of departure or amending personal information (Rebook flight). After this, only cancellation is permitted (Undo booking), in case the passenger ultimately decides not to proceed with boarding (Board flight).

The behavioral constraints specifying how the tasks can be carried out are the following:

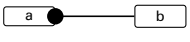
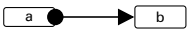
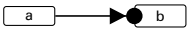
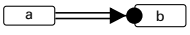
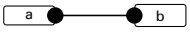
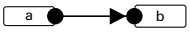
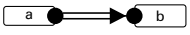
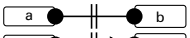
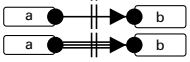
1. Init(Start booking)
2. AtMostOne(Start booking)
3. Alt.Precedence(Start booking, Provide data)
4. Alt.Succession(Provide data, Pay)
5. Alt.Succession(Pay, Complete transaction)
6. Alt.Succession(Pay, Complete booking)
7. Alt.Precedence(Complete booking, Rebook flight)
8. NotSuccession(Check-in, Rebook flight)
9. Alt.Precedence(Check-in, Board flight)
10. NotSuccession(Board flight, Undo booking)
11. NotSuccession(Undo booking, Provide data)

2.2. Declare in process mining

Process mining is a family of techniques used to analyze and improve business processes by extracting insights from the so-called *event logs* [15] generated during the execution of those processes [16]. Declarative process discovery is a branch of process mining concerning the discovery of declarative process models from event logs. This is the group of techniques we deal with in this paper, as *we propose a new visual notation that we demonstrate improves the current output of these process mining techniques, which is primarily based on the original graphical representation of* Declare.

In the area of declarative process discovery with models expressed using Declare, the work [17] presents an unsupervised algorithm for the discovery of Declare models. In [18], the authors present an algorithm for the discovery of Declare constraints that makes the discovery

Table 1
DECLARE constraints.

Template	Explanation	Examples				Notation
Existence templates						
EXISTENCE(n, a)	Activity a occurs at least n times in the trace					
PARTICIPATION(a) ≡ EXISTENCE(1, a)	a occurs at least *once*	✓ bcac	✓ bcaac	× bcc	× c	
ABSENCE($m+1$, a)	a occurs at most m times					
ATMOSTONE(a) ≡ ABSENCE(2, a)	a occurs at most *once*	✓ bcc	✓ bcac	× bcaac	× bcacaa	
INIT(a)	a is the *first* to occur	✓ acc	✓ abac	× cc	× bac	
END(a)	a is the *last* to occur	✓ bca	✓ baca	× bc	× bac	
Relation templates						
RESPONDEDEXISTENCE(a, b)	If a occurs in the trace, then b occurs as well	✓ bcaac	✓ bcc	× caac	× acc	
RESPONSE(a, b)	If a occurs, then b occurs after a	✓ caacb	✓ bcc	× caac	× bacc	
ALTERNATERESPONSE(a, b)	Each time a occurs, then b occurs afterwards, before a recurs	✓ cacb	✓ abcacb	× caacb	× bacacb	
CHAINRESPONSE(a, b)	Each time a occurs, then b occurs immediately afterwards	✓ cabb	✓ abcab	× cacb	× bca	
PRECEDENCE(a, b)	b occurs only if preceded by a	✓ cacbb	✓ acc	× ccbb	× bacc	
ALTERNATEPRECEDENCE(a, b)	Each time b occurs, it is preceded by a and no other b can recur in between	✓ cacba	✓ abcaacb	× cacbba	× abbabcb	
CHAINPRECEDENCE(a, b)	Each time b occurs, then a occurs immediately beforehand	✓ abca	✓ abaabc	× bca	× baacb	
Mutual relation templates						
COEXISTENCE(a, b)	If b occurs, then a occurs, and vice-versa	✓ cacbb	✓ bcca	× cac	× bcc	
SUCCESSION(a, b)	a occurs if and only if it is followed by b	✓ cacbb	✓ accb	× bac	× bcca	
ALTERNATESUCCESSION(a, b)	a and b if and only if the latter follows the former, and they alternate each other in the trace	✓ cacbab	✓ abcabc	× caacbb	× bac	
CHAINSUCCESSION(a, b)	a and b occur if and only if the latter immediately follows the former	✓ cabab	✓ ccc	× cacb	× cbac	
Negative relation templates						
NOTCOEXISTENCE(a, b)	a and b never occur together	✓ cccbbb	✓ ccac	× accbb	× bcac	
NOTSUCCESSION(a, b)	a can never occur before b	✓ bbcaa	✓ cbbca	× aacbb	× abb	
NOTCHAINSUCCESSION(a, b)	a and b occur if and only if the latter does not immediately follows the former	✓ acbacb	✓ bbaa	× abcab	× cabc	

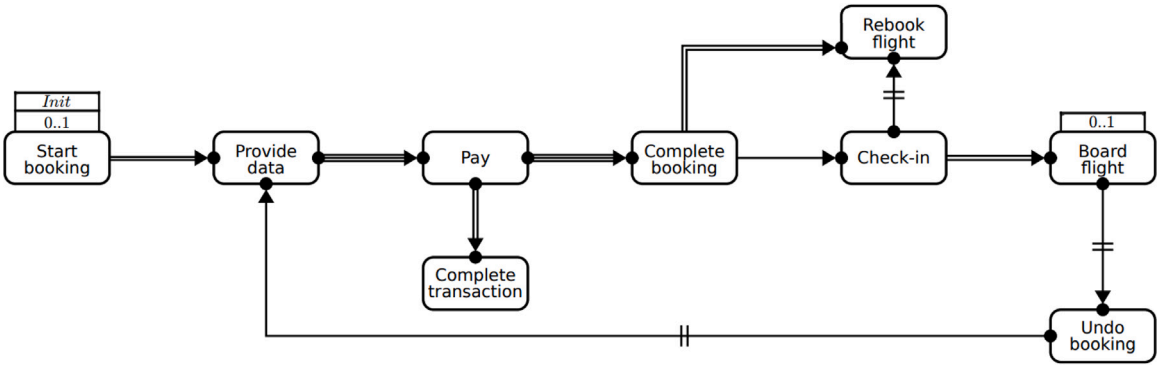


Fig. 1. The DECLARE model of a flight booking process.

efficient via the use of a knowledge base containing information about temporal statistics about the (co-)occurrence of tasks within the log.

Another approach for the discovery of DECLARE models is described in [19]. The presented technique is based on the translation of DECLARE

templates into SQL queries on a relational database instance, where the event log has previously been stored. An Evolutionary Declare Miner that implements the discovery task using a genetic algorithm was presented in [20]. In [21], the authors investigate how to leverage Model Learning algorithms [22] for the automated discovery of deterministic finite state automata representing DECLARE templates from event logs.

The approaches mentioned above specify the discovered process models leveraging the original graphical representation of DECLARE [3]. One of the contributions of this paper is the implementation of a web application for designing declarative process models using EASYDECLARE. The application also implements a functionality for the discovery of EASYDECLARE models. The discovery approach implemented in the application follows the one outlined in [17], providing users with an efficient and accessible way to extract declarative process models from event logs. This web-based tool offers an intuitive interface for process discovery, demonstrating how process discovery techniques can benefit from the understandability of EASYDECLARE. This language enhances usability of process discovery algorithms, making it easier for users to explore and comprehend complex workflows.

2.3. Designing and developing visual notations

The use of visual notations is extensive throughout engineering and information technology (IT), where shapes and patterns aim to represent processes, e.g., BPMN, flowcharts, etc. Developing a visual language's syntax requires an informative approach that considers several key parameters of visual communication, such as the level of cognitive processing required to understand and use the notation, the shapes used to communicate the processes, and the conventions followed. By documenting this design approach, the final design is justified, and insights can be gained into aspects that effectively and efficiently communicate processes and those that do not.

Several frameworks provide principles for researchers developing or evaluating visual notations within a scientific context. For example, the Cognitive Dimensions of Notations (CDs) framework defines 13 dimensions that describe the structure of cognitive artifacts [23], while the Semiotic Quality (SEQUAL) framework proposes general qualities for models and modeling languages, organized along the semiotic ladder (i.e., the scale 'physical', 'empirical', 'syntactic', 'semantic', 'pragmatic', and 'social') [24].

However, one of the more notable examples, and one that is extensively used, is *The Physics of Notations theory* by Moody [13] (PoN). PoN provides the foundation for many studies either as a way to inform the development of new visual notations or to examine the effectiveness of existing ones. This theory presents a framework for how visual notations are constructed and processed by the human mind, drawing on theories from communication, graphic design, semiotics, visual perception, and cognition. Moody's seminal work emphasizes the meanings of graphical symbols and how they are defined by mapping them to the constructs they represent [13]. Moody asserts that visual variables define a set of atomic building blocks for constructing visual notations, where these variables provide the grammar for the notation. Therefore, the choice of visual variables should not be arbitrary but should consider the conventions of symbols currently used or accepted within the field, follow a logical process, and be easily understood by those using and interpreting the notation. This consideration is particularly important because each variable has properties that can make it more suitable for encoding certain types of information over others.

Based on this information, Moody presents nine principles for designing cognitively effective visual notation, as shown in Fig. 2. The following definitions are described in [25]:

1. Semiotic clarity: There should be a 1:1 correspondence between semantic constructs and graphical symbols.

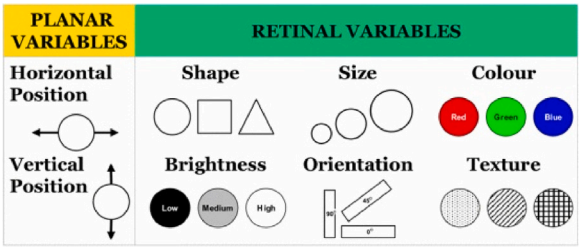


Fig. 2. Moody's visual variables [13].

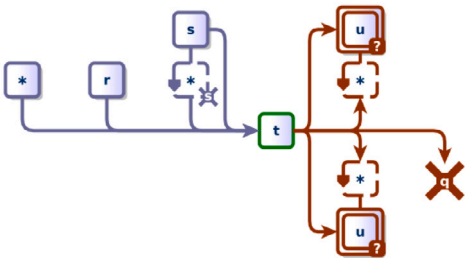
2. Perceptual discriminability: Symbols should be clearly distinguishable from one another.
3. Semantic transparency: Use symbols whose appearance suggests their meaning.
4. Complexity management: Include explicit mechanisms for dealing with complexity.
5. Cognitive integration: Include explicit mechanisms to support the integration of information from different diagrams.
6. Visual expressiveness: Use the full range and capacities of visual variables.
7. Dual coding: Use text to complement graphics.
8. Graphic economy: Keep the number of different graphical symbols cognitively manageable.
9. Cognitive fit: Use different visual dialects for different tasks and/or audiences.

Moody outlines three key properties that should be considered in the development of visual notation:

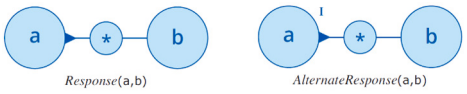
- **Level of organization**: Each variable can encode a certain level of information. For example, the use of color (e.g., red/green) can indicate success or error.
- **Capacity**: While each variable has infinite variations, only a finite number can be easily interpreted by the human mind (perceptible steps). For example, orientation has an infinite number of possible values (angles) but only four perceptible steps (its capacity or length).
- **Efficiency**: The order in which variables are processed (e.g., in parallel, sequentially, etc.).

In addition, Moody indicates approaches for dealing with excessive graphic complexity, such as reducing semantic and graphic complexity and increasing visual expressiveness.

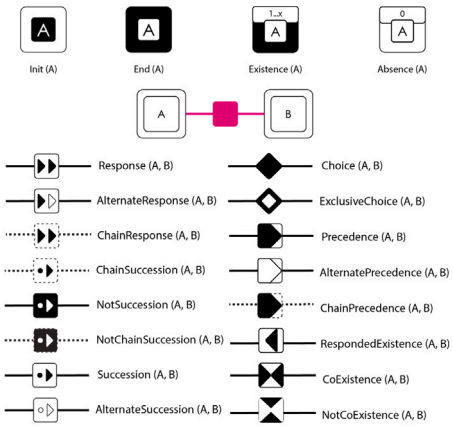
Further emphasizing Moody's work, both Popescu and Wegmann [26] and Diamantopoulou and Mouratidis [27] explore the use of Moody's nine principles. Diamantopoulou and Mouratidis [27] assert that the design rationale for developing visual notation is often absent in the design of visual notations. They infer that this could be due to the fact that the description and reporting of the language are oriented towards science rather than art or design or that researchers consider visual notations as being informal and, therefore, analyze the notations based on their semantics, potentially undermining or undervaluing the role of design. Similarly, Popescu and Wegmann [26] and Genon, Heymans, and Amyot [28] also suggest using the set of nine principles defined by Moody [13] to evaluate how effectively modeling languages communicate their intended messages. Lastly, others have investigated systematic approaches to applying PoNs (e.g., [25,29]), where such approaches, such as Van Der Linden, Zamansky, and Hadar [25], propose a systematic framework for applying PoNs that focuses on guiding designers to make their design choices explicit and grounded in evidence and requirements for their notation.



(a) Example local view of the activity a including RespondedExistence(t, u) and NegationResponse(t, q) from Di Ciccio et al. [5].



(b) Response(a, b) and AlternateResponse(a, b) templates from Hanser et al. [6].



(c) Verto templates presented in [12].

Fig. 3. Alternative notations for declarative constraints.

2.4. Alternative notations for declarative constraints

Efforts to improve the interpretability of declarative models have been diverse, with some graphical notations being proposed. Di Ciccio et al. [5] introduced a graphical notation designed for declarative processes. Their approach includes two complementary views of a model: a local and a global view. The local view focuses on one activity at a time, providing detailed insights. The global view offers an overview of the entire model, showcasing the interconnections and flow. This notation uses a bi-dimensional drawing where time is represented on the ordinates (y-axis) and implication on the abscissa (x-axis). The thickness of the boundaries around the boxes representing activities indicates their repeatability, cf. Fig. 3(a). This dual view aims to enhance the user's understanding of the process by visually differentiating between temporal and causal relationships.

Conversely, Hanser et al. [6] propose an alternative notation that challenges the traditional Declare notation by using circles instead of squares. This notation incorporates cursors to indicate sequential relations between activities, with inwards and outwards cursors at the end of arcs. Optional activities are represented by smaller circles placed in the middle of the arc, often accompanied by an asterisk, indicating that additional constraints may allow further activities to be executed in between.

Hanser's notation represents an evolution of the declarative templates initially presented by van der Aalst et al. [3]. It offers a fresh perspective on designing process notations by revisiting and modifying existing elements, aiming to provide a clearer and more intuitive visualization of constraints and sequences within declarative models. Figs. 3(a) and 3(b) illustrate these notations, showing the Response(a, b) and AlternateResponse(a, b) templates encoded using the respective styles proposed by Di Ciccio et al. and Hanser et al.

Finally, in 2018, Ferro et al. [12] presented Verto, which included restyling the original Declare constraints to align them with the state-of-the-art design principles for visual notations. The list of Declare constraints represented with the Verto notation is shown in Fig. 3(c). These visual representations [5,6,12] underscore the distinct approaches to modeling and interpreting declarative processes using Declare, each with its own set of advantages, limitations and enhancements.

3. Design

In this section, we discuss the rationale behind the design of easyDeclare. Specifically, in Section 3.1 we first present the results of a preliminary analysis conducted with three experts in declarative modeling, aimed at gaining insights into their perspectives on the understandability of the original Declare notation and, where possible, relating these insights to the PoN dimensions. Then, based on this analysis, in Section 3.2 we describe the design of easyDeclare.

3.1. Preliminary analysis

The design of easyDeclare was primarily informed by a preliminary analysis of the understandability of the original Declare notation [3] from the viewpoint of domain experts. The initial design step involved conducting three in-depth, semi-structured interviews [30] with domain experts. These experts included two professors and one practitioner, all of whom possessed at least five years of experience using and teaching the Declare language. All interviewees were external to the pool of authors of this paper, ensuring independence in the evaluation. Each interview was conducted separately (two online and one in person) by two of the paper's authors, namely an expert in declarative modeling and another in visual analytics and PoN dimensions. During these interviews, experts were asked to evaluate the understandability of the Declare notation and discuss its strengths and weaknesses. The interviews focused on four major themes:

- *Perceived ease of reading*, how participants approached interpreting Declare models and whether they found the notation intuitive;
- *Interpretation of constraints*, how well participants understood the meaning of individual Declare constraints and their interaction in models;
- *Challenges in model comprehension*, difficulties encountered when interpreting models of different sizes and complexity;
- *Perceived ambiguities and usability issues*, cases where participants reported confusion, alternative interpretations, or a need for clarification.

None of the three experts was familiar with the PoN dimensions [13] (researchers and practitioners often lack awareness of these dimensions [31]). The interview data were processed to analyze the experts' utterances concerning understandability, with the aim to link them

(when possible) to the PoN dimensions. Additionally, we also conducted a review of the literature on the understandability of declarative notations and of DECLARE in particular. This allowed us to triangulate the empirical findings from the interviews with established knowledge and prior empirical results.

For the *semiotic clarity* principle, there should be a $1:1$ correspondence between semantic constructs and graphical symbols. DECLARE activities are depicted as rectangles, while constraints are combinations of lines (for alternate and chain alternatives), dots (for activations), and arrowheads (for ordering). While this principle is generally observed, the mapping between the number of lines used for constraints (two lines for alternate templates, three for chain templates) was deemed ineffective by Expert 2. This confusion required users to "double-check the correct mapping" if they had not recently used DECLARE. Furthermore, Haisjackl et al. [8] revealed that aspects that appear similar in imperative and declarative process modeling languages at a graphical level, while having different meanings, caused considerable difficulties.

The *perceptual discriminability*, dictating that symbols should be clearly distinguishable, is affected by the association between the number of lines and template variations, particularly in complex models where identifying variations relying solely on the number of lines was not proficient, according to Expert 3.

Figl et al. [9] performed empirical experiments on the *semantic transparency* of DECLARE, suggesting that the representation of constraints as arrows might be semantically perverse to users. The analysis performed by Trinh et al. [10] on DCR came to similar conclusions, highlighting that arrows are generally associated with sequences rather than causality. In line with these findings, all experts indicated that arrow-based representations can mislead users into interpreting constraints as control-flow dependencies. In particular, Expert 1 emphasized the importance of employing visual metaphors that better reflect the declarative nature of the models, favoring symbols or spatial groupings that communicate relationships and conditions rather than sequential order.

Regarding *complexity management*, DECLARE does not explicitly address the complexity of large models. The exploratory study of Haisjackl et al. [8] raised potential concerns, showing that subjects could understand a single constraint well, but it is challenging to handle a combination of constraints. Additionally, DECLARE does not include explicit mechanisms for *cognitive integration* or constructs to integrate data from different models.

Concerning *visual expressiveness*, the shape is the primary visual variable, while the position of dots indicates activations. The other variables are not used, and text (*dual coding*) is not used for either annotations or hybrid coding. The principle of *graphic economy* is observed, with a limited amount of symbols (lines, dots, bars, arrowheads) used to encode binary constraints and rectangles to denote activities. Expert 1 noted that after users "grasp and get used to the association between the symbols and their meaning, and acquire pattern recognition skills, the representations of constraints become natural".

About *cognitive fit*, Experts 2 and 3 observed that imperative paradigms align more intuitively with the way new users process information. This supports the findings of Pichler et al. [11], who showed that novice users understand imperative languages better than declarative ones. This suggests that Declare might have a lack of cognitive fit for novice users.

The analysis concluded that Declare meets some of Moody's principles, but significant room for improvement exists, especially concerning perceptual discriminability, semantic transparency, and complexity management.

3.2. Design of easyDECLARE

Based on this analysis, we designed a new notation, named easyDECLARE. The new notation addresses the identified limitations by simplifying the mapping of concepts and aiming at reducing the cognitive load required to interpret models with many templates. The templates in the new notation are constructed by composing a few elementary visual elements complying with the graphic economy. These elements are designed to be intuitive and easy to remember, reducing the need for extensive memorization.

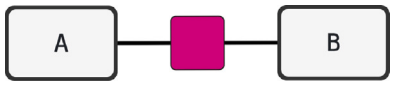


Fig. 4. Binary constraint between two activities (A and B) as defined by easyDECLARE visual notation. A single glyph over the arc represents the constraint. Each constraint primitive has its own glyph; in this figure, the purple box is a placeholder for that.

The DECLARE and Hanser et al. relation templates are constructed by applying visual means distributed throughout the arc joining the two activities. The templates for constraints implying causality (response, precedence, and succession) are shown as cursors (at the end of the arc in DECLARE, at both extremes in Hanser et al.), while activations are shown at the ends of the arc (as circles in DECLARE). In DECLARE, the number of lines represents the variations of the constraint (plain, alternate, chain), while negation is represented with two vertical lines in the middle of the arc. In Hanser et al. the cursors are positioned internally or externally to the activity, depending on the activations, and are hollow to represent the negation. The fact that information is spread can be an issue when templates are used in models that have many constraints. When information is spread across different parts of a diagram, users need to mentally connect these parts to understand the relationship. This can be cognitively demanding. For this reason, we decided to depict the binary templates with a single glyph on the arc defined according to the constraint it represents, as shown in Fig. 4. The arc is a plain line and does not contain any other visual means (e.g., arrows) except the glyph.

By consolidating all the information into a single symbol, users can interpret the information more quickly and with less mental effort. This improves the readability of complex models, as users do not need to look at multiple locations to gather the information. When information is located at the ends of arcs, there is a higher chance of misinterpreting or missing parts of the information, especially in complex diagrams. A single symbol minimizes this risk by presenting all the relevant information in one place, thereby enhancing accuracy. This choice is consistent with the Proximity Compatibility Principle [32], which claims that *the elements that need to be integrated and processed together should be in close proximity*. Using a single symbol to represent the constraint makes all necessary data spatially close, making it easier to process the information. However, this design choice may also introduce potential trade-offs. In particular, when models are mined and contain a large number of constraints, relations may occur between events that are positioned far apart in the layout. In such cases, the glyph may be visually distant from its source and target events, potentially increasing the reliance on the subject's memory and, consequently, cognitive load. Furthermore, in highly dense models with a variety of different constraints, the use of arcs with identical shapes could reduce perceptual discriminability and create difficulties in distinguishing between constraint types.

To foster semiotic clarity, shapes are mapped $1:1$ to different concepts: horizontal rectangles represent activities, squares represent relation templates, triangles (arrows) represent sequentiality, and vertical bars represent activations. All the binary templates are shaped as squares, with the only variation occurring with CHOICE(a, b) and EXCLUSIVECHOICE(a, b), where the shapes are diamonds. The reason for this is that the diamond is associated with choice or decision in both Flowcharts and UML.

The use of color is another area that, in many ways, offers an additional layer of implicit information. For example, the color red

(a) ChainSuccession(a, b) **(b)** NotChainSuccession(a, b)

Fig. 5. An example of a ChainSuccession(a, b) and a NotChainSuccession(a, b). The white background (a) and black background (b) on the constraint symbol discriminate between a positive and a negative relation.

suggests an error, while the color green denotes success. In addition, the use of color also contains accessibility limitations in situations where users may suffer from a vision impairment (e.g., color blindness). For these reasons, the only colors that are used for easyDeclare are black and white. The first reason is its *ease of use* and *accessibility*. By using black and white, we mitigate accessibility issues, as the contrast between the two is enough to be seen by a majority of users. The second reason is related to printing. If a user intends to print their easyDeclare models, the use of black and white allows it to be clearly interpreted regardless of the printer type that is used (i.e., monochrome or color). Templates are generally encoded through a black glyph over a white background, the opposite for negative constraints. This color differentiation allows users to easily tell apart models such as ChainSuccession(a, b) and NotChainSuccession(a, b), as shown in Fig. 5, where the former is white and the latter is black.

The use of arrows to indicate direction is a well-established convention in graph theory and diagramming. However, previous studies [9, 10] have shown that in the context of declarative languages, arrows can be misinterpreted, as users tend to associate them with sequential order rather than with causality. This suggests that their use may inadvertently lead to misunderstandings about the type of relations being represented.

The proposed notation does not use arrows directly (having no additional symbols on the line). However, we decided to represent the order between activities in the response, precedence, and succession constraints by inserting an arrowhead inside the glyph of the constraint. This representation should ensure that the order of activities can be correctly interpreted even in complex models where they can be positioned counter-intuitively (e.g., right to left, bottom to top)

The activations are represented as vertical bars inside the glyph representing the constraint. The bars are positioned according to the role of the activities; for example, Response(a, b) has a bar on the left of the arrowhead because the activity activating it is the first one, while Precedence(a, b) has a bar on the right because the template it activates is the one that follows the other one. Distinctive examples are CoExistence(a, b) and RespondedExistence(a, b); the symbol of the former consists of the two bars of the activations, while the latter has the left activation bar and a smaller bar on the right, thus recalling the direction of the template.

Finally, variations of relation templates are encoded as annotations with letters embedded in the symbol: *A* for "Alternate" variations, and *C* for "Chain" variations. Let us remark that this choice may affect semantic transparency, especially for non-native English speakers, as suggested by López and Simon [4] in their analysis of DCR. This may also affect perceptual discriminability due to the lack of visual distance, but we chose this as we did not want to introduce additional elements for the graphic economy principle.

Templates. Following the aforementioned primitives, we defined existence and relation templates. Activities are encoded with white rectangles, although a light gray background can optionally be used to improve contrast. The name of each activity is written inside the rectangle, ensuring clear identification and easy readability. Existence constraints are represented as labels attached to the top or the bottom of the activity rectangles. These labels contain text that specifies the unary constraint. Each template has a fixed position over the activity rectangle to leverage preattentive visual properties. The differentiation

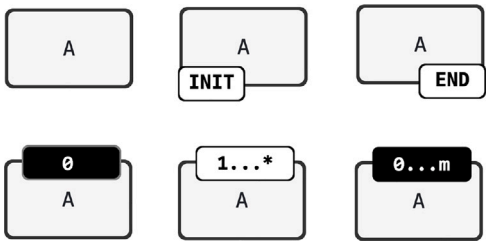


Fig. 6. easyDeclare notation for existence constraints. First row: plain activity *A*, Init(*A*), End(*A*). Second row: Absence(1, *A*), Existence(1, *A*), Absence($m+1$, *A*).

of placement makes the constraints easily recognizable at a glance. Fig. 6 shows how existence constraints are represented.

Relation templates are represented by a glyph positioned in the middle of the arc connecting two activities. The arc is plain, without any additional visual elements, ensuring that the focus remains on the glyph representing the constraint. Each type of relation template is associated with a unique glyph, which is shown in 7.

Example 3.1 (*A Flight Booking Process Model*)**.** The scenario presented in Example 2.1, which illustrates the process a flight passenger follows, is now shown in Fig. 8 using the easyDeclare notation.

4. Alignment with PoN principles

To assess the design of our notation against established guidelines, we conducted an expert evaluation focused on Moody's Physics of Notations (PoN) [13]. The goal of this activity was to obtain a critical perspective on how well easyDeclare aligns with the PoN principles, validating its design choices. For this evaluation, we interviewed two PoN experts external to the pool of the paper's authors, ensuring independence in the evaluation. The two experts are a full professor and an associate professor, both with extensive experience in human–computer interaction and information visualization. In addition, to position our notation in relation to previous works, experts were asked to evaluate Declare and the Hanser et al. notation [6] (in this section, simplified as Hanser).

4.1. Methodology

We conducted in-depth, semi-structured interviews [30] with the two selected experts in a joint session. We chose this method to encourage a reflective discussion among the experts and support a deeper investigation of the design characteristics of each notation.

We followed a fixed order of analysis: first Declare, then Hanser, and finally easyDeclare. Before each analysis, we introduced the notation, illustrating its syntax, semantics, and visual representation. For every PoN dimension, the experts were initially asked to independently identify and report all relevant aspects of the analyzed notation that related to the specific dimension. Subsequently, the experts jointly examined their observations and aligned on a shared interpretation. After reaching a consensus, the experts were asked to jointly assign a coverage rating for that dimension using a three-level Likert scale: *not satisfied at all*, *partially satisfied*, or *fully satisfied*.

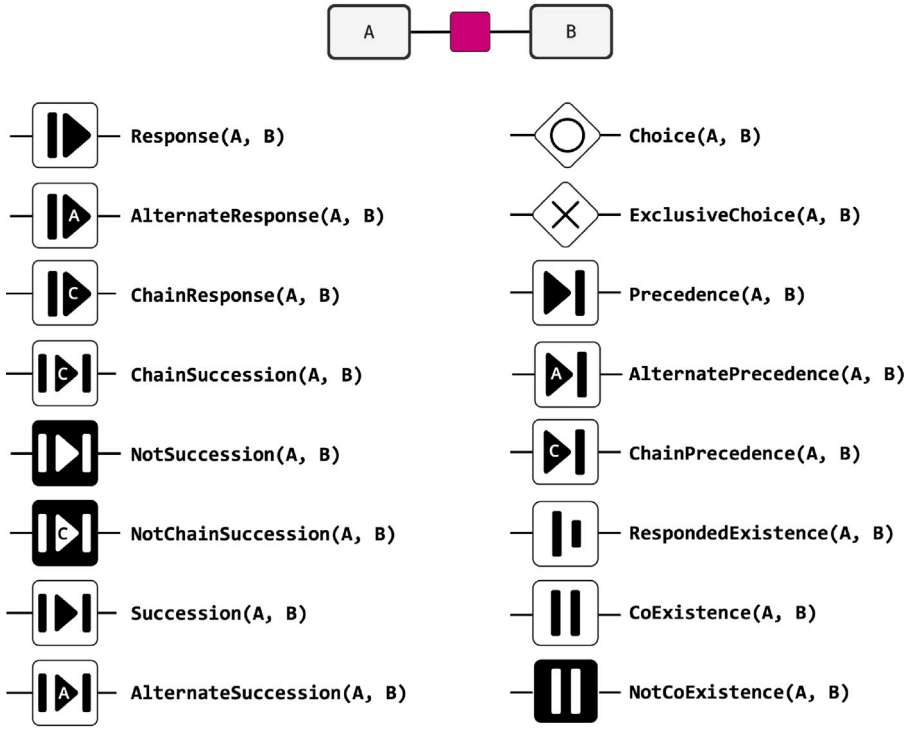


Fig. 7. easyDeclare graphical notation for binary constraints. Template icons are available at https://github.com/easyDeclare/templates.

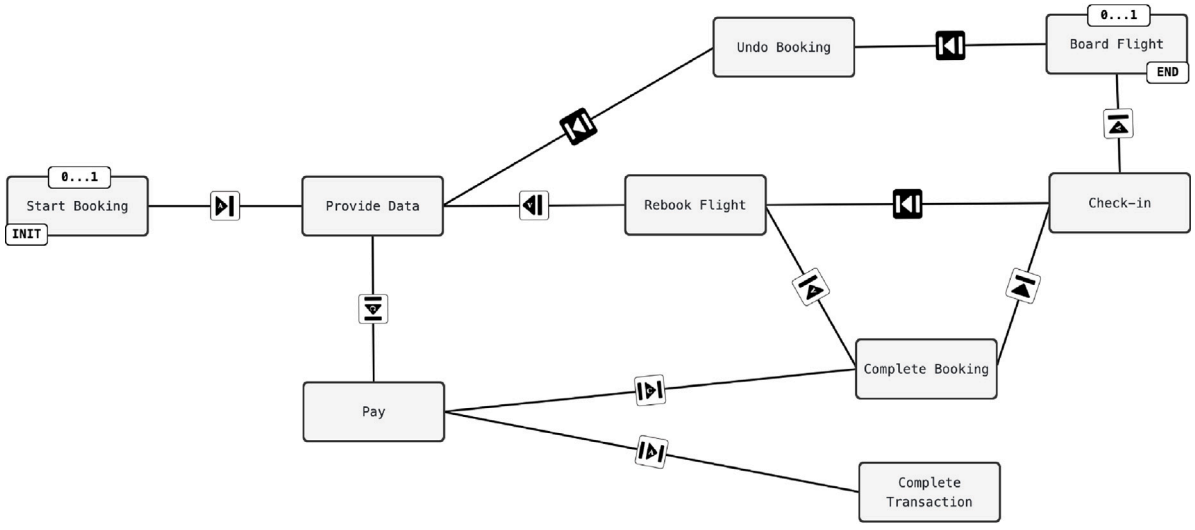


Fig. 8. The easyDeclare model of a flight booking process, same as Example 2.1 (Fig. 1).

4.2. Results

The results of the interviews are reported in detail for each PoN principle in the following, while Table 2 reports a summary of coverage. Table 3 summarizes coverage and rationales. Overall, easyDeclare appeared to improve the coverage of the PoN principles with respect to Declare and Hanser, performing better in *semiotic clarity*, *perceptual discriminability*, *semantic transparency*, and *visual expressiveness*. At the same time, both Declare and Hanser have been noticed to behave similarly in terms of PoN principles application.

Semiotic clarity. The experts judged that only easyDeclare fully satisfies the principle of semiotic clarity, as it consistently employs a one-to-one mapping between visual symbols and semantic concepts. In contrast, both Declare and Hanser were evaluated as only partially satisfying this dimension. Declare employs similar visual encodings (lines, dots, and bars) across different templates, which can lead to ambiguity. Hanser has been noted for an improvement over Declare, by introducing a mixture of visual forms, such as circles and different line styles; however, it still does not clearly distinguish each concept with a unique symbol.

Perceptual discriminability. This principle is fully supported only by easyDeclare, which combines shape and text (for binary constraints) and positional coding (for unary constraints) to clearly distinguish templates. Declare and Hanser only partially satisfy this principle, with experts noting a low visual distance between similar symbols or the overuse of minute differences, such as line orientation or slight variations in arrowheads.

Table 2
Assessment of DECLARE, HANSER, and EASYDECLARE notations against Moody's Physics of Notations (PoN) dimensions.

PoN dimension	DECLARE	HANSER	EASYDECLARE
Semiotic clarity	P	P	S
Perceptual discriminability	P	P	S
Semantic transparency	P	P	S
Complexity management	N	N	N
Cognitive integration	N	N	N
Visual expressiveness	P	P	S
Dual coding	N	N	P
Graphic economy	S	S	S
Cognitive fit	N	N	N

Coverage legend: S satisfied, P partially satisfied, N not satisfied.

Semantic transparency. DECLARE and HANSER were rated as only partially satisfying semantic transparency: both employ arrowheads and directional lines that can wrongly suggest sequentiality instead of causality, as also confirmed by [9]. In contrast, EASYDECLARE appears to satisfy the principle of semantic transparency because it avoids misleading arrows and incorporates mnemonic glyphs with additional letters (A/C) that help differentiate symbols.

Complexity management. None of the three notations addresses complexity management. The experts highlighted that the notations share the goal of representing activities and constraints, and they offer no solutions oriented to support large and complex models, as also noted by [8].

Cognitive integration. None of the notations support cognitive integration.

Visual expressiveness. This principle is defined as the number of visual variables used in a notation. According to Moody, the visual variables are eight: horizontal position, vertical position, size, brightness, color, texture, shape, and orientation. Moody recommended using as many visual variables as possible: the more visual variables are used, the higher the visual expressiveness. According to the two experts, DECLARE uses 3 visual variables: horizontal position, vertical position, and shape. HANSER uses 4 visual variables: horizontal position, vertical position, brightness (where white color is used for negation, as well as for binary constraints, making it not a unique mapping), and shape. EASYDECLARE employs 5 visual variables: horizontal position, vertical position, brightness (where white and black distinguish between positive and negative relations), shape, and orientation. Visual expressiveness was rated highest for EASYDECLARE, which uses five out of eight available visual variables. The absence of size, color, and texture was not considered critical for this type of notation, pushing the experts to judge EASYDECLARE as fully satisfying the principle of visual expressiveness.

Dual coding. DECLARE and HANSER do not use dual coding solutions. Conversely, the experts rated dual coding as partially satisfied by EASYDECLARE. The notation uses both text and positions for unary constraints. For binary constraints, variations of relation templates are encoded as annotations with letters embedded in the symbol: A for "Alternate" variations, and C for "Chain" variations. Experts noted that this does not comply with the dual coding principle because no other visual variable is used in conjunction with the text. Since dual coding is not present in all constraints, the experts rated this principle as partially satisfied. Additionally, this choice may affect semantic transparency, especially for non-native English speakers [4].

Graphic economy. All three notations were rated as satisfying the principle of graphic economy. DECLARE uses a very small symbol set (lines, dots, bars, arrowheads). However, the experts emphasized that this economy can come at the cost of clarity, where this minimalism leads to visual ambiguity. HANSER uses a small symbol set, including circles for activities and binary constraints, as well as arrowheads and diamondheads for binary constraints. In EASYDECLARE the symbol set is slightly larger, including rectangles, bars, triangles, diamonds, circles, and squares. Although this symbol set is larger than the others, its usage remains justifiable due to its improved clarity.

Cognitive fit. None of the notations were considered to satisfy the cognitive fit principle. According to the experts, all three rely on a single visual vocabulary, offering no customization for different user groups (e.g., novices vs. experts), a limitation previously highlighted for Declare by Pichler et al. [11].

5. Evaluation

We designed a controlled experiment to evaluate the proposed graphical notation. First, we describe the experiment's design, then present and discuss the obtained results.

5.1. Design

The design of the experiment relies on the Method Evaluation Model (MEM) [33], collecting performance-based and perception-based metrics to evaluate the likelihood of acceptance of EASYDECLARE. We designed a comparative experiment in which subjects had to perform tasks using the original graphical representation of DECLARE (cf. Table 1), and the one we propose, EASYDECLARE (cf. Figs. 6 and 7).

A graphical notation for declarative process modeling supports two elementary tasks:

- *Template encoding* – given a template, depict it through its graphical representation;
- *Template decoding* – given the graphical representation of a template, identify it correctly.

In its practical application, it typically supports higher-level tasks built on the elementary ones:

- *Model encoding* – given a model, depict it through its graphical representation;
- *Model decoding* – given the graphical representation of a model, comprehend its meaning.

For encoding tasks, a template (or a model) was presented to the subject, who had to select its representation from five alternatives; conversely, for decoding tasks, the subject had to choose the textual representation corresponding to the graphical representation of a template (or a model) given as input.

According to the MEM, the efficacy in performing a task is defined as the quality of the results (*effectiveness*) and the effort required (*efficiency*) in achieving them. For the experiment tasks, effectiveness corresponds to the correctness of the result, while efficiency can be defined as the ratio between the effectiveness and the time spent to perform the task.

The experiment targets novice users without prior knowledge of graphical notations for declarative process modeling. Therefore, we decided to include the evaluation of the learning process of notations by subjects in the experiment. In the first phase of the experiment, the subjects had to perform elementary tasks alternated with increasingly detailed explanations of the notation. The results collected in this phase contribute to evaluating the semantic transparency and learnability of the notations.

Some empirical studies [34,35] suggest that problem-solving tasks can be used to measure understandability. We, therefore, used model encoding and decoding to evaluate it. We designed the experiment as a balanced single-factor experiment with repeated measurement [36,37].

Each subject experimented first with one notation and then with the other. Between the two stages of the experiment, a break was made to

Table 3
Detailed assessment of DECLARE, HANSER, and EASYDECLARE notations against Moody's Physics of Notations (PoN) dimensions.

PoN dimension	Notation	Coverage and Rationale	
Semiotic clarity	DECLARE	P	Many constraints look similar; 1:1 concept-symbol not always met.
	HANSER	P	Shapes are mapped to different concepts: circles represent activities; text annotations for unary constraints; a combination of solid and dashed lines, circles, and arrowheads for binary constraints. 1:1 concept-symbol not always met.
	EASYDECLARE	S	Shapes are mapped 1:1 to different concepts: horizontal rectangles represent activities, squared glyphs represent relation binary constraints with triangles (arrows) for sequentiality, and vertical bars for activations.
Perceptual discriminability	DECLARE	P	Low visual distance among constraint symbols, especially when they differentiate only by the number of lines.
	HANSER	P	Using inward and outward cursors and their combination to represent causality and activations are difficult to distinguish.
	EASYDECLARE	S	Glyph forms + text (binary) or positional coding (unary) clearly distinguish templates.
Semantic transparency	DECLARE	P	Arrows suggest sequences rather than causality and can be semantically perverse [9].
	HANSER	P	Arrows suggest sequences rather than causality. Inward and outward c, placed inside or outside the activity circles, have different meanings but can be easily confused.
	EASYDECLARE	S	No arrows, mnemonic glyphs and letters (A/C) help to differentiate symbols.
Complexity management	DECLARE	N	Large-model complexity: not addressed explicitly [8].
	HANSER	N	Not addressed.
	EASYDECLARE	N	Not addressed.
Cognitive integration	DECLARE	N	No constructs for integrating data from different models.
	HANSER	N	No constructs for integrating data from different models.
	EASYDECLARE	N	No constructs for integrating data from different models.
Visual expressiveness	DECLARE	P	Usage 3 out of 8 visual variables: horizontal position, vertical position, and shape.
	HANSER	P	Usage 4 out of 8 visual variables: horizontal position, vertical position, brightness, and shape. White color is used for negation, but also for binary constraints, so it is not a unique mapping.
	EASYDECLARE	S	Usage 5 out of 8 visual variables: horizontal position, vertical position, brightness, shape, and orientation. White and black distinguish between positive and negative relations.
Dual coding	DECLARE	N	No usage.
	HANSER	N	No usage.
	EASYDECLARE	P	Unary: text+position. Binary: glyphs with embedded letters for variants (alternate/chain); no usage on other binary constraints.
Graphic economy	DECLARE	S	Very small symbol set (lines, dots, bars, arrowheads), but at the cost of clarity.
	HANSER	S	Small symbol set: circles for activities and binary constraints, arrowheads and diamondheads for binary constraints.
	EASYDECLARE	S	Expanded glyph vocabulary (rectangles, bars, triangles, diamonds, circles, squares), keep limited.
Cognitive fit	DECLARE	N	Single vocabulary; no audience-specific variants. Lack of cognitive fit for novice users [11].
	HANSER	N	Single vocabulary; no audience-specific variants.
	EASYDECLARE	N	Single vocabulary; no audience-specific variants.

Coverage legend: S satisfied, P partially satisfied, N not satisfied.

reduce carryover effects. An instance of the experiment is generated by randomly choosing the templates and the models used for the tasks. Let us remark that the templates and models a user has experienced for one notation are different from those experienced for the other to reduce learning biases. To minimize order effects (e.g., learning, fatigue), we ensured that different participants experienced the conditions in a different order. The experiments lasted 40.83 min on average with a standard deviation of 11.85 (not including the 30-minute break between the two stages). For each instance created for a user, another is created for another one in which the notations are swapped so that the order in which they are presented and the variability of the templates affect the result the least. This instance is divided into two parts, A and B, corresponding to the two notations.

Each part was structured as follows: As the first step, the subject had to perform 4 tasks of *Template decoding* (TD_1) and 4 tasks of *Template encoding* (TE_1) without a prior explanation of the notation to evaluate its *semantic transparency*.

Afterwards, subjects were provided with the list of templates and relative graphical representations (E_1), which they could consult for a maximum of 3 min. Then, they had to perform another 4 tasks of *Template decoding* (TD_2) and 4 tasks of *Template encoding* (TE_2).

The subjects were then provided with the list of templates and relative graphical representations and a brief explanation of the rationale behind the notation (E_2), which they could consult for as long as they deemed necessary.

Finally, they had to perform the last block of tasks composed of 4 tasks of *Template decoding* (TD_3), 4 tasks of *Template encoding* (TE_3), 4 tasks of *Model decoding* (MD_1), and 4 tasks of *Model encoding* (ME_1). Fig. 9 shows an example of a *Model decoding* task. Abbad-Andaloussi et al. [38] introduced tailored metrics to measure the complexity of declarative models. The $Size$ of a declarative process model is the sum of its activities and constraints, its $Density$ is the maximum ratio of constraints over activities in the weakly connected components of the graph, and the $Separability$ is the number of weakly connected components over the number of activities and constraints in the model. The models used in the experiment have $Size = 7$, $Density = 4/3$, $Separability \in [1/7, 2/7]$. Templates were chosen randomly, while the models were created manually to ensure their consistency. Specifically, the tested models were generated with the following guarantees: (i) each constraint was covered in at least one model; and (ii) the same template (even with different activities) could not appear more than once in the same model.

We collected the score (i.e., 10 if the answer is correct, 0 otherwise) and elapsed time for each performed task, and the time devoted by each subject to the two explanations.

Additionally, at the end of each part, the subjects had to answer a questionnaire regarding their perception of the graphical notation adapted from Abrahão et al. [39] visible in Table 4.

The users had to answer 16 questions with a five-level Likert scale, ranging from "*strongly disagree*" to "*strongly agree*". The answers are then mapped into a numerical value ranging from $[0, 10]$. The questions are intended to estimate three perception-based variables: Perceived Ease of Use (PEOU), i.e., the degree to which a person believes that using a particular technology or system will be free from effort; Perceived Usefulness (PU), i.e., the extent to which a person believes that using a specific technology or system will provide some beneficial outcome;

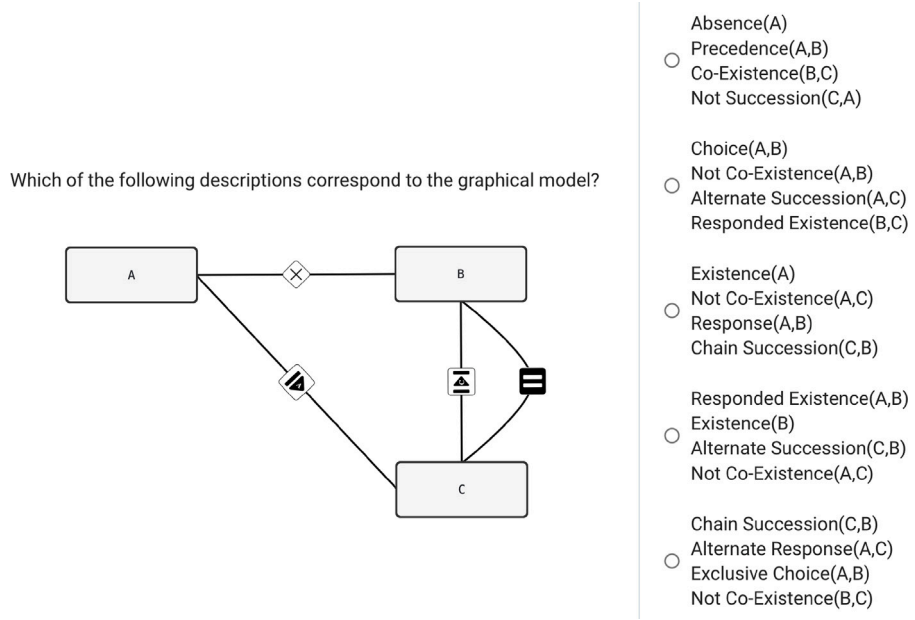


Fig. 9. Example of a *Model decoding* task for the easyDeclare notation where the correct answer is the last one.

Table 4
Questionnaire used to assess the user perception of the graphical notation. *Source:* Adapted from Abrahão et al. [39].

Item	Statement
PEOU1	The graphical notation is simple and easy to use.
PEOU2	Overall, it was easy to understand what the graphical templates represent (Template Decoding).
PEOU3	Overall, it was easy to understand what the graphical models represent (Model Decoding).
PEOU4	Overall, it was easy to represent the templates with the graphical notation (Template Encoding).
PEOU5	Overall, it was easy to represent the models with the graphical notation (Model Encoding).
PEOU6	The graphical notation is easy to learn.
PU1	Overall, I found the graphical notation to be useful.
PU2	Overall, I think this graphical notation provides an effective way of describing declarative templates.
PU3	I believe this graphical notation would reduce the time required to model declarative processes.
PU4	I believe this graphical notation is useful for modeling business processes in complex environments.
PU5	I believe that the models obtained with this graphical notation are organized, clear, concise and non-ambiguous.
PU6	I believe this graphical notation has enough expressiveness to represent declarative processes.
PU7	Using this graphical notation would improve my performance in describing complex business processes.
ITU1	I would use this graphical notation to specify complex business processes.
ITU2	It would be easy for me to become knowledgeable in using this graphical notation.
ITU3	I would recommend the use of this graphical notation to describe complex business processes.

and Intention to Use (ITU), i.e., the user's motivation or plan to employ a particular technology or system in the future. We acknowledge that a similar analysis exploring the perceived usefulness and ease of use of two declarative process languages other than Declare, namely DCR and CMMN, was conducted by Jalali in [40].

5.2. Hypotheses

According to the Goal-Question-Metric (GQM) template [41], the experimentation goal is to: *Analyze* the declarative process modeling graphical notations easyDeclare and Declare, *for the purpose of* evaluating their effectiveness and efficiency, *with respect to* their semantic transparency, learnability, understandability, ease of use, usefulness, and intention to use, *from the point of view of* novice and non-expert users.

The evaluation measures 3 performance-based variables (semantic transparency, learnability, and understandability) and 3 perception-based variables (PEOU, PU, and ITU). Table 5 shows the independent and dependent variables of the study and how they were calculated.

The results are analyzed with a *paired t-test* to check whether there is a statistically significant difference between the average performance with the two notations ($p - value < 0.05$). Specifically, we formulate the following hypotheses:

Semantic transparency.

- **H1**$_0$ Users without prior knowledge of easyDeclare and Declare have the same effectiveness and efficiency in decoding templates represented with the two notations.
- **H2**$_0$ Users without prior knowledge of easyDeclare and Declare have the same effectiveness and efficiency in encoding templates with the two notations.

Learnability.

- **H3**$_0$ Users without prior knowledge of easyDeclare and Declare have the same effectiveness and efficiency in learning to decode templates represented with the two notations.
- **H4**$_0$ Users without prior knowledge of easyDeclare and Declare have the same effectiveness and efficiency in learning to encode templates with the two notations.
- **H5**$_0$ Users without prior knowledge of easyDeclare and Declare take the same amount of time to memorize the two notations.

Table 5
Independent and dependent variables.

Type	Subtype	Name	Measure
Independent variables	Graphical notations	DECLARE easyDECLARE	
Dependent variables	Performance variables	Score Time	Correct answer: 10, Wrong answer: 0 Seconds
	Performance-based variables	Semantic Transparency effectiveness Semantic Transparency efficiency Understandability effectiveness Understandability efficiency Learnability effectiveness Learnability efficiency Learnability time	Score of TD_1, TE_1 Score/Time of TD_1, TE_1 Score of MD_1, ME_1 Score/Time of MD_1, ME_1 Score of $(TD_1, TD_2, TD_3), (TE_1, TE_2, TE_3)$ Score/Time of $(TD_1, TD_2, TD_3), (TE_1, TE_2, TE_3)$ Time of E_1, E_2
	Perception-based variables	Perceived Ease of Use Perceived Usefulness Intention to Use	Mean of PEOU1, …, PEOU6 from Table 4 Mean of PU1, …, PU7 from Table 4 Mean of ITU1, …, ITU3 from Table 4

Understandability.

H6$_0$ Users have the same effectiveness and efficiency in decoding models represented with the two notations.
H7$_0$ Users have the same effectiveness and efficiency in encoding models with the two notations.

Perception-based variables.

H8$_0$ The Perceived Ease of Use is the same for the two notations.
H9$_0$ The Perceived Usefulness is the same for the two notations.
H10$_0$ The Intention to Use is the same for the two notations.

Subjects. The subjects who participated in the experiment are 31 students with prior knowledge of the syntax and semantics of declarative process modeling templates but not of their graphical notations. We had to discard one result due to technical problems that occurred during the experiment, reducing the number of valid experiments to 30.

The use of students as subjects might affect the experiment's external validity; however, they are relatively close to the population of interest, being the next generation of professionals, and can be considered representative of novice and non-expert users of declarative process modeling [42,43]. We anonymously collected the results and did not inform subjects of which notation we proposed to limit the experimental bias that may affect the experiment's internal validity.

Subjects were 15 males and 15 females, 29 Master students and 1 Ph.D. student in Engineering in Computer Science.

5.3. Analysis

Semantic transparency. Regarding semantic transparency (Fig. 10), almost no significant differences emerge between the two notations. The only difference appears in the template decoding effectiveness (Fig. 10(a)), where using easyDECLARE ($\mu_E = 6.4$) is greater than using DECLARE ($\mu_D = 5.0$). This difference is not statistically significant in template efficiency (Fig. 10(b)) for both decoding and encoding.

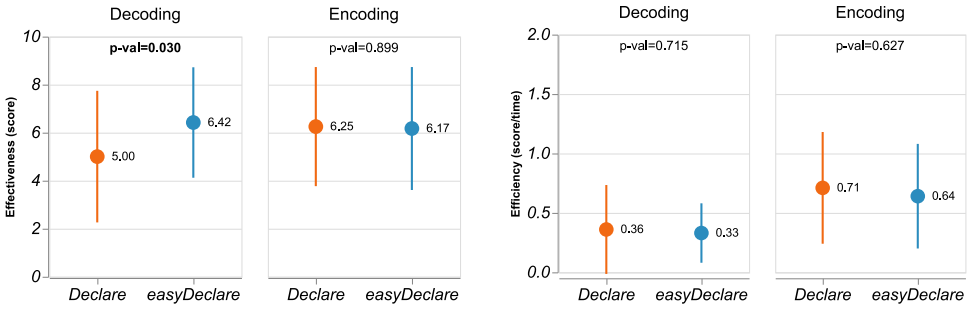
Results: **H1**$_0$ is partially rejected for the difference in the average effectiveness. However, no significant differences were observed in the efficiency of decoding. This is because the users were more effective using easyDECLARE but at the cost of more time taken to respond. **H2**$_0$ is accepted, thus no differences were found between the two notations for encoding.

Learnability. Subsequently, subjects were provided with a list of templates and their relative graphical representations. The data show no statistically significant differences in the time the subjects spent on the two notations (Fig. 11(e).E1). Subsequent decoding tasks show no significant differences in efficiency and effectiveness. Conversely, template encoding shows significant differences between the two notations, with easyDECLARE having better performance both in terms of effectiveness (Fig. 11(a) $\mu_E = 9.7$, $\mu_D = 8.8$) and efficiency (Fig. 11(b) $\mu_E = 1.6$, $\mu_D = 1.3$).

The subjects were then provided with a list of templates and their relative graphical representations, along with a brief explanation of the rationale behind the notation. The time spent on the two notations is different in this case (Fig. 11(e).E2), with that devoted to easyDECLARE ($\mu_E = 32.4$) significantly lower than that devoted to DECLARE ($\mu_D = 61.4$). The last block of template decoding and encoding tasks does not show significant differences. The only difference appears in decoding efficiency (Fig. 11(d)), with the efficiency of easyDECLARE ($\mu_E = 1.45$) higher than the efficiency with DECLARE ($\mu_D = 1.16$).

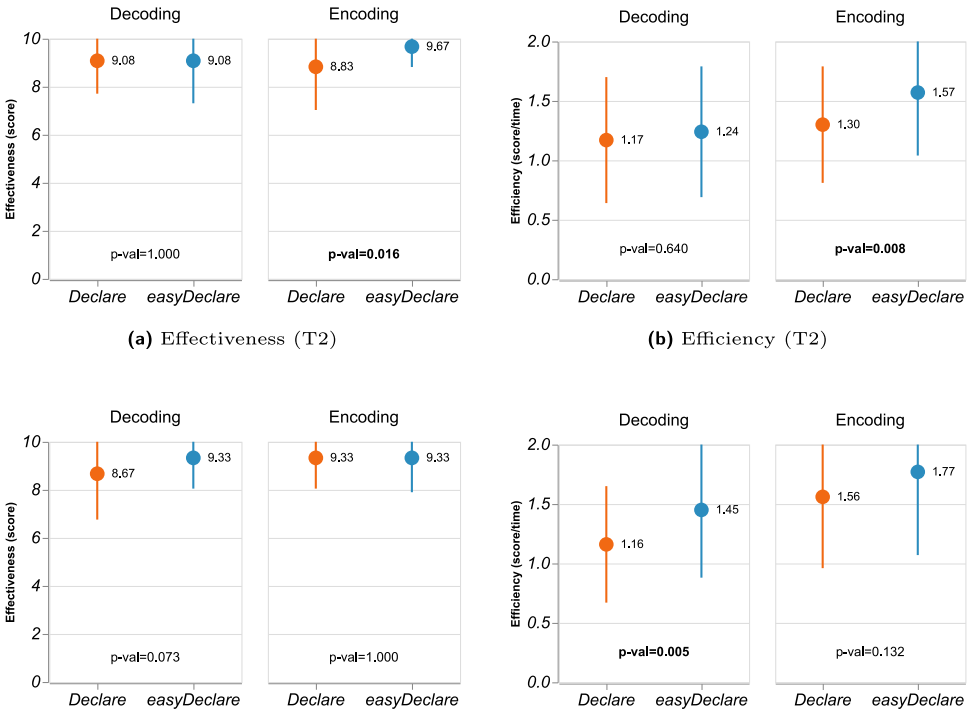
Results: **H3**$_0$ is almost fully accepted; however, the third time the subjects had to decode the templates, they were more efficient using easyDECLARE. **H4**$_0$ is partially rejected; after the first explanation of the notations, subjects showed that they were more effective and more efficient using easyDECLARE. However, these differences are not significant after further explanation of the notations. The analysis of the time spent analyzing the notations seems to confirm these results. While the time spent on the first explanation does not show significant differences, the time spent on the second explanation is significantly higher for DECLARE, suggesting that the subjects were less confident with this notation. **H5**$_0$ is thus partially rejected.

Understandability. The tasks dealing with the decoding and encoding of entire models are where the most significant differences emerged (see Fig. 12). Decoding performance with easyDECLARE was better than with DECLARE, both for effectiveness and efficiency. With easyDECLARE, subjects scored on average $\mu_E = 8.2$ for effectiveness and $\mu_E = 0.29$ for efficiency, whereas with DECLARE the performance was much lower ($\mu_D = 2.9$ for effectiveness and $\mu_D = 0.11$ for efficiency). Similar performances were observed in model encoding. The effectiveness of easyDECLARE ($\mu_E = 9.3$) was much higher than that of DECLARE ($\mu_D = 4.7$); similarly, the efficiency of the former ($\mu_E = 0.46$) was higher than the efficiency of the latter ($\mu_D = 0.27$). The exploratory study of Haisjackl et al. [8] observed a discrepancy in the understanding of DECLARE similar to that found in this study, showing that subjects could understand a single constraint well, but that it is challenging for them to handle a combination of constraints.



(a) Effectiveness (T1) **(b)** Efficiency (T2)

Fig. 10. Semantic transparency. Dots show means; vertical bars denote mean ± SD. Reported p-values compare the two notations. Numerical results are in Fig. 14. easyDeclare statistically performs better than Declare in decoding effectiveness. For the other measures, almost no differences emerge between the two notations.

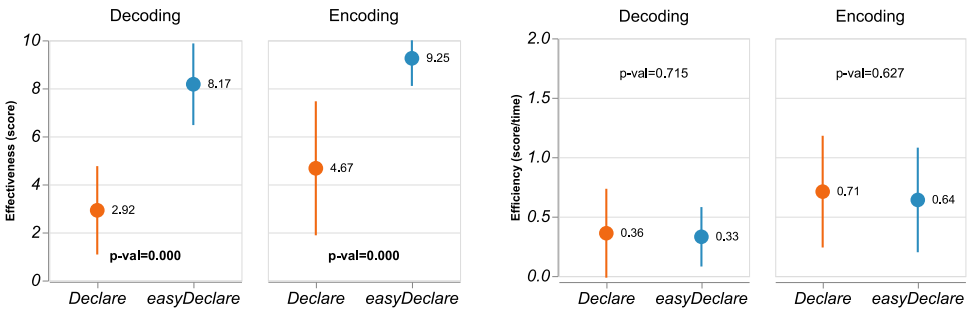


(a) Effectiveness (T2) **(b)** Efficiency (T2)

(c) Effectiveness (T3) **(d)** Efficiency (T3)

(e) Time

Fig. 11. Learnability. Dots show means; vertical bars denote mean ± SD. Reported p-values compare the two notations. Numerical results are in Fig. 14. easyDeclare statistically performs better than Declare in Task T2 for both encoding effectiveness and efficiency; in task T3 for decoding efficiency; and in E2 learning time.



(a) Effectiveness **(b)** Efficiency

Fig. 12. Model Understandability. Dots show means; vertical bars denote mean ± SD. Reported p-values compare the two notations. Numerical results are in Fig. 14. easyDeclare statistically performs better than Declare in effectiveness.

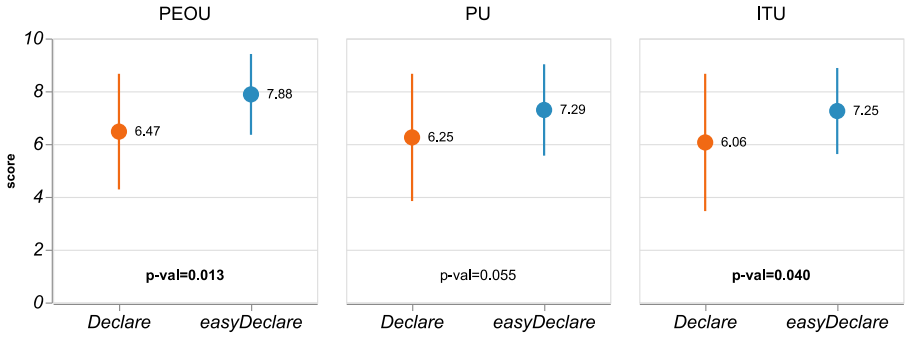


Fig. 13. Perception-based variables. Dots show means; vertical bars denote mean ± SD. Reported p-values compare the two notations. Numerical results are in Fig. 14. easyDeclare statistically performs better than Declare in Perceived Ease of Use (PEOU) and Intention to Use (ITU). No statistical difference appears in perceived Usefulness (PU).

Results: Both hypotheses on understandability are rejected. The differences shown are substantial for both decoding and encoding models in terms of effectiveness ($\mathbf{H6}_0$) and efficiency ($\mathbf{H7}_0$). Although they cannot be considered conclusive, these results show that the users were consistently more proficient using easyDeclare both in model encoding and decoding tasks.

Perception-based variables. Perception-based variables show some differences between the two notations (see Fig. 13). More specifically, the PEOU of easyDeclare ($\mu_E = 7.88$) is higher than that of Declare ($\mu_D = 6.47$). Similarly, the ITU of easyDeclare ($\mu_E = 7.25$) is higher than that of Declare ($\mu_D = 6.06$), while the PU does not show significant differences.

Results: $\mathbf{H8}_0$ is rejected, suggesting that the Perceived Ease of Use of easyDeclare is higher than that of Declare. Similar considerations arise for the Intention to Use, as $\mathbf{H10}_0$ is rejected. Conversely, the Perceived Usefulness does not show significant differences, as $\mathbf{H9}_0$ is confirmed.

5.4. Threats to validity

In the following, we discuss aspects threatening the validity of our study following the categorization proposed by Wohlin et al. [44].

Internal validity. Abbad-Andaloussi et al. [38] defined different metrics to evaluate the complexity of declarative process models and investigated their relationship with cognitive load. An important observation is that the models we used in our study did not have different levels of complexity between them. This implies that the resulting analysis relies on a specific cognitive load, and we have not analyzed whether the results change as the complexity of the models changes (and thus as the required cognitive load changes, cf. [37] for an extensive study of this matter). Furthermore, not all users have tested all templates for both notations. The templates used in the study were extracted uniformly, and we verified before conducting the experiment that all templates were examined by at least one user in each notation for decoding or encoding. The pairwise generation of the traces for the experiment ensures that the same templates (and patterns) were evaluated for both notations, but not that all templates have the same level of coverage.

Construct validity. Regarding construct validity, different factors threaten the validity of the study. To evaluate semantic transparency, we evaluated whether users were able to correctly associate templates with their graphical representations. This binary way of measuring semantic transparency can only be considered an approximation of the definition given by Moody who noted how it can vary on a scale from semantic immediate to semantic perverse. The experiment does not include specific measures or questionnaires designed to probe the subjects' subjective understanding of the semantic transparency of individual symbols or their position on a "perverse to immediate" scale. Another concern for construct validity (and indirectly to conclusion validity) is that the results seem to be affected by the ceiling effect [45]. This seems especially relevant for T2 and T3 in which many participants were able to achieve maximum scores (in terms of effectiveness) with both notations, thus reducing the statistical significance of the results. An additional aspect to consider is that we measured understandability through model encoding and decoding tasks. This is not necessarily representative of all aspects related to understandability, and there may be comprehension tasks where the effectiveness of notation differs. An example might be tasks where the user is interested in identifying all the constraints activated by a specific activity for which we expect Declare to perform much better

	T1		T2		T3	
p-value	**0.030**		1.000		0.073	
Notation	e	D	e	D	e	D
Mean	6.42	5.00	9.08	9.08	9.33	8.67
Std. Dev.	2.30	2.74	1.77	1.37	1.28	1.91

(a) Decoding Effectiveness

	T1		T2		T3	
p-value	0.899		**0.016**		1.000	
Notation	e	D	e	D	e	D
Mean	6.17	6.25	9.67	8.83	9.33	9.33
Std. Dev.	2.56	2.48	0.85	1.80	1.43	1.28

(b) Encoding Effectiveness

	T1		T2		T3	
p-value	0.715		0.565		**0.005**	
Notation	e	D	e	D	e	D
Mean	0.33	0.36	1.24	1.17	1.45	1.16
Std. Dev.	0.25	0.37	0.55	0.53	0.57	0.49

(c) Decoding Efficiency

	T1		T2		T3	
p-value	0.627		**0.008**		0.256	
Notation	e	D	e	D	e	D
Mean	0.64	0.71	1.57	1.30	1.77	1.56
Std. Dev.	0.44	0.47	0.53	0.49	0.70	0.60

(d) Encoding Efficiency

	Decoding		Encoding	
p-value	**0.000**		**0.000**	
Notation	e	D	e	D
Mean	8.17	2.92	9.25	4.67
Std. Dev.	1.70	1.84	1.15	2.79

(e) Models Effectiveness

	Decoding		Encoding	
p-value	**0.000**		**0.002**	
Notation	e	D	e	D
Mean	0.29	0.11	0.46	0.27
Std. Dev.	0.16	0.10	0.24	0.22

(f) Models Efficiency

	E1		E2	
p-value	0.589		**0.033**	
Notation	e	D	e	D
Mean	114.6	106.9	32.4	61.4
Std. Dev.	55.8	51.7	23.2	56.3

(g) Explanation Time

	PEOU		PU		ITU	
p-value	**0.013**		0.055		**0.040**	
Notation	e	D	e	D	e	D
Mean	7.88	6.47	7.29	6.25	7.25	6.06
Std. Dev.	1.53	2.19	1.73	2.41	1.63	2.60

(h) Perception Variables

Fig. 14. Analysis results across tasks and measures. Each panel reports, for easyDeclare (e) and Declare (D), the mean and standard deviation (SD). The top row shows the between-notation p-value (bold when $p < 0.05$).

by representing the activations directly on the activity. We did not investigate explicitly whether the two notations have differences in supporting the identification of inconsistencies in the models [46]. The models used had no inconsistencies, and the study does not include tasks for moving from a model to an execution trace that could have supported this investigation.

External validity. Using novice users in experiments can threaten external validity [47] because the results may not generalize to real-world scenarios where experienced users use the notation. This creates an interaction of selection and treatment threat, meaning the findings may be specific to novices and not applicable to skilled practitioners. Experienced users may have different problem-solving strategies, cognitive loads, or efficiency levels, leading to different outcomes in practice. Specifically, their perception of the Perceived Usefulness has limited significance as they have no experience with the use of these instruments. In order to test the learnability in a controlled way, it was necessary to establish a single learning process, which may not, however, correspond to the actual learning process of a notation. This type of assessment can provide insights into the differences in the learning of the two notations but the results cannot be considered conclusive. Furthermore, the experts who examined the three notations in terms of their alignment with the PoN dimensions are not experts in declarative languages. They were introduced to the notation by the authors of the paper, and this may have introduced a bias into their assessment.

Conclusion validity. The relatively low number of users who carried out the experiment may influence the significance of it. Analysis of the results also showed that a statistically significant difference was not measured for all variables. For semantic transparency, we observed a significant difference only in decoding effectiveness. In the variables measured to test learnability, we observed some significant differences but not enough to ensure that the whole learning process of easyDeclare is undoubtedly more effective or more efficient than Declare. The evaluation of understandability showed the greatest differences; however, we must emphasize that the results may be influenced by fatigue accumulated during the experiment. Users conducted two consecutive sessions in which they evaluated one notation first, then the other. The overall duration of the experiment and the fact that they had to learn and use one notation first and then the other may have influenced the results and induced a carryover effect.

6. EDD – EasyDeclare Designer

We developed EDD – *EasyDeclare Designer* – a tool to support the creation and editing of models based on our proposed graphical notation. Furthermore, it supports process discovery activities from event logs. EDD is an open-source web application implemented in JavaScript, along with a Python backend, and is freely available along with its source code at https://github.com/easyDeclare/EDD. Further

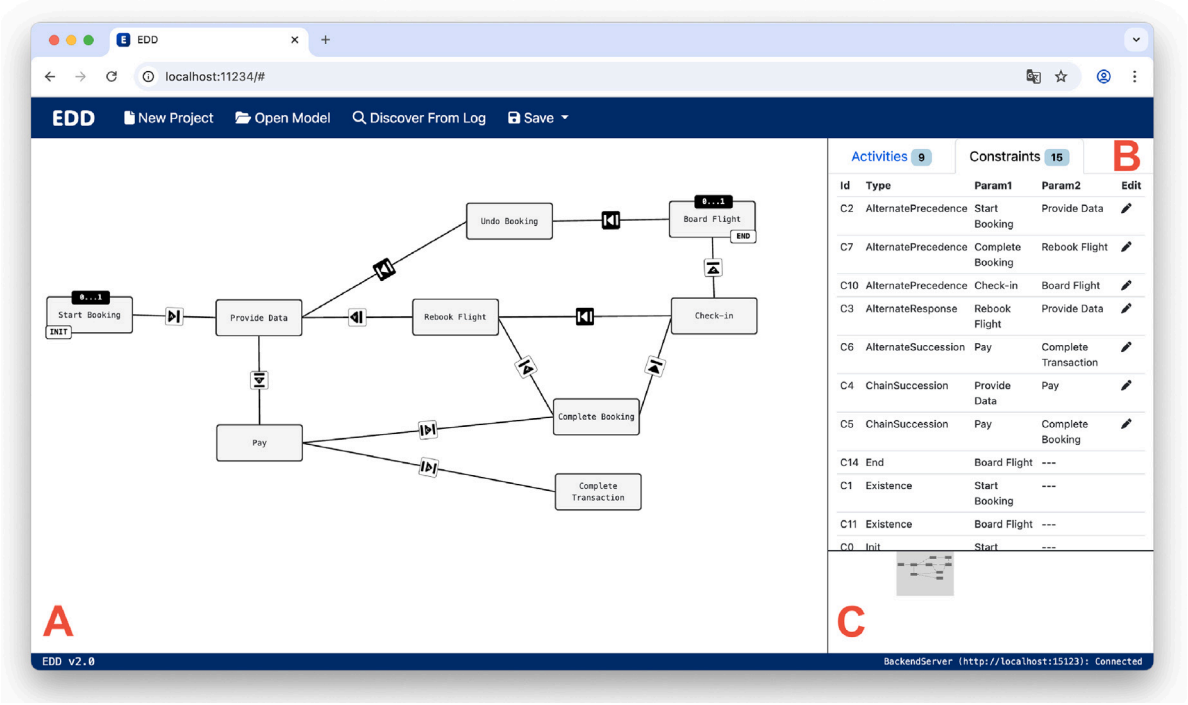


Fig. 15. An overview of the EDD interface. It is composed of three panels: (A) *Canvas Pane*, (B) *Details Pane*, and (C) *Overview Pane*.

details about the tool, including features and technical specifications, are reported in Section 6.1.

To evaluate the usability of EDD, we conducted a user study using the System Usability Scale (SUS), a common approach for measuring the usability of systems, providing a quantitative score that reflects the overall user experience and assesses their perceived usability. The results of the SUS indicated a high usability level, achieving an A+ rating. Details about this evaluation are reported in Section 6.2.

6.1. Tool overview

The interface of EDD has been designed to be simple and efficient. As shown in Fig. 15, it is composed of three panels: (A) *Canvas Pane*, (B) *Details Pane*, and (C) *Overview Pane*.

The main pane is the *Canvas* (A) on the left side of the interface, which shows the model. This pane also offers an interactive interface for model creation and manipulation. Users can create activities by double-clicking on an empty area of the canvas. To create binary constraints between activities, users can mouse over the border of an activity and drag it to another activity. Although the tool automatically arranges activities and constraints on the canvas to maintain an organized layout, users can manually adjust the positions by dragging these elements across the canvas, allowing for a custom layout. Furthermore, the canvas supports zoom and pan functionalities, supporting users to manage models of various sizes.

On the right side of the interface is the *Details Pane* (B), which provides an overview of the existing activities and constraints in the model. This pane also displays details about each activity and constraint, allowing the editing of their properties.

In the bottom right corner of the interface, there is the *Overview Pane* (C). This pane shows the overview of the entire model, highlighting which part is currently visible in the canvas, following the focus+context visual paradigm. This feature enables users to maintain a comprehensive view of the entire model, even as they zoom in and out on the canvas to work on different areas.

To allow for the interoperability of the tool, we implemented existing standards for the input and output of model files, such as *decl* [48] and *RuM* [49]. Additionally, we defined a new input/output format named *edj*, which is based on JSON and contains information about activities, constraints, and layout. This format is detailed in the tool repository.

An additional feature of EDD is the capability to support process discovery activities, enabling users to derive declarative process models directly from event logs. To achieve this, the tool accepts event logs in the well-known XES format [50]. By leveraging *Declare4Py* [51], a Python library for DECLARE model discovery, EDD can analyze the provided event logs and extract the underlying EASYDECLARE model, i.e., the declarative constraints describing the process traced in the logs. The discovery results benefit from the usability of EASYDECLARE (demonstrated in the following section), making it easier for users to interpret and interact with the extracted declarative models.

6.2. Usability evaluation

To evaluate the usability of EDD, we conducted a user study using the System Usability Scale (SUS) [52,53]. The SUS is a questionnaire composed of ten questions designed to assess the perceived usability of a system. The user, after having performed tasks on the system, is required to answer the 10 questions with a value on a five-level Likert scale, ranging from "*strongly disagree*" to "*strongly agree*". The answers are then mapped into a numerical value ranging from $[0, 10]$, and their sum becomes the SUS score assigned to the system, which ranges from $[0, 100]$. Additionally, Lewis and Sauro [54] defined a scale mapping the SUS score into 11 grades from higher to lower: [*A+, A, A−, B+, B, B−, C+, C, C−, D, F*].

The ten questions of the SUS are reported in the following:

Q1. I think that I would use this system frequently;
Q2. I found the system unnecessarily complex;
Q3. I thought the system was easy to use;
Q4. I think that I would need the support of a technical person to be able to use this system;
Q5. I found the various functions in the system were well integrated;
Q6. I thought there was too much inconsistency in this system;

Q7. I would imagine that most people would learn to use this system very quickly;
Q8. I found the system very awkward to use;
Q9. I felt very confident using this system;
Q10. I need to learn a lot of things before I could get going with this system.

The user study involved 18 researchers and practitioners in the process mining field, covering a diverse range of expertise. The participants included 1 associate professor, 3 assistant professors, 10 Ph.D. students, 2 master students, 1 bachelor student, and 1 process mining consultant. Participants' average age is 29. Notably, none of the selected users had conflicts of interest with the authors of this paper, ensuring an unbiased evaluation of the tool's usability.

6.2.1. Methodology

Participants were first asked to evaluate their expertise in three areas: (a) declarative process modeling and related graphical notations, (b) imperative process modeling and related graphical notations, and (c) BPMN and related graphical notations. This was done using a five-level Likert scale ranging from "*not competent*" to "*highly competent*". Next, we briefly reviewed the basic concepts of Declarative Process Modeling and illustrated our proposed visual notation. We then presented EDD, explained its functionalities, and allowed the participants ten minutes to familiarize themselves with the tool.

To test the system functionalities, we designed two tasks (T1 and T2) for the users to complete. After completing the tasks, participants were asked to fill out the SUS questionnaire. Finally, participants were invited to submit any additional comments and feedback on the tool anonymously through an online form, ensuring fair responses and providing valuable insights for future improvements.

T1. The first task is designed to familiarize users with the system by creating a simple model from scratch. Users were asked to model an Emergency Room procedure as follows:

1. Open the tool and create a new project.
2. Add three new activities named *Registration*, *Admission*, and *Triage*.
3. Add a Init unary constraint to the activity *Registration*.
4. Add an Existence constraint with cardinality 0...1 to the activity *Registration*.
5. Add a binary constraint AlternateResponse($Registration, Admission$).
6. Add a binary constraint Succession($Registration, Triage$).
7. Add a binary constraint Precedence($Triage, Admission$).
8. Save the model to a local file.

T2. The second task involves modifying an existing and more complex model (shown in Fig. 15), which represents a booking procedure for flight tickets. Users were asked to:

1. Open the "flight.edj" file in the tool.
2. Add a Init unary constraint to the activity *StartBooking*.
3. Add a End unary constraint to the activity *BoardFlight*.
4. Add a new activity named *UndoBooking*.
5. Add a binary constraint NotSuccession($BoardFlight, UndoBooking$).
6. Add a binary constraint NotSuccession($UndoBooking, ProvideData$).
7. Transform the Choice($ProvideData, Pay$) into ChainSuccession($ProvideData, Pay$).
8. Invert the order of activities in the AlternateSuccession($CompleteTransaction, Pay$) to AlternateSuccession($Pay, CompleteTransaction$).
9. Save the model to a local file.

The correct model, after the required modifications, is shown in Fig. 8.

6.2.2. Results

Participants' expertise was assessed using a five-level Likert scale, where "*not competent*" was encoded as 0 and "*highly competent*" as 10. The expertise levels were as follows (with μ and σ being the average score and the standard deviation, respectively):

- Declarative process modeling: theory $\mu = 5.56$, $\sigma = 3.16$; related graphical notations $\mu = 5.42$, $\sigma = 3.56$.
- Imperative process modeling: theory $\mu = 5.52$, $\sigma = 3.56$; related graphical notations $\mu = 5.42$, $\sigma = 3.76$.
- BPMN: theory $\mu = 7.36$, $\sigma = 2.5$; related graphical notations $\mu = 7.08$, $\sigma = 2.88$.

Overall, the SUS results demonstrate a highly positive evaluation of the tool's usability, with an average grade of A+, an average score of $\mu = 87.64 \pm 3.48$ (confidence level: 95%), and a median of 90. This indicates that users generally found the system very usable. The standard deviation of $\sigma = 7.54$ suggests relatively low variability in scores, meaning that most users had a similar usability experience with the tool. Detailed question and score results are shown in Fig. 16.

Interestingly, question Q1 — *I think that I would use this system frequently* — received the lowest median among all questions. However, this does not necessarily indicate an issue with the tool's usability. When participants were asked to comment on their responses to Q1, many explained that their roles do not involve frequent creation or editing of models, thus reducing the need for regular system use.

Participants provided several positive comments and valuable suggestions for improving the tool. Users generally appreciated its visual design and usability. For example, one user commented, "*Very nice visual tool, with clear feedback and high usability level*". Another noted, "*In general, I found the tool very simple to use*".

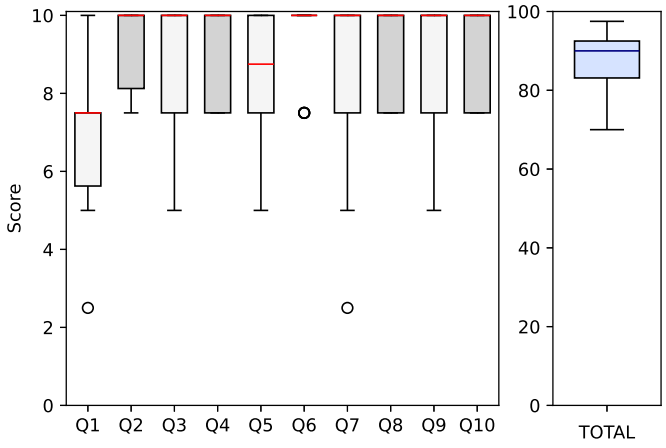
Despite these positive remarks, users also offered constructive feedback for enhancements. One suggestion was to "*add support for multiple projects open at the same time in different tabs*", which would improve multitasking capabilities. Another user suggested, "*It would be helpful to double-click on an activity to rename it*", indicating a potential improvement in interaction design. Additionally, implementing keyboard shortcuts, such as "*clicking on an activity and pressing the delete key to remove the activity*", was also recommended.

7. Conclusion

Providing interpretable and easy-to-understand process models is a precondition for developing effective visual analytics tools in process mining [1]. In this paper, we introduced and evaluated a new graphical notation, easyDeclare, designed to improve the interpretability of declarative process discovery results. Traditional process modeling languages, while effective for specialists, often pose comprehension challenges for non-experts due to their complexity and technical nature [55].

The design of easyDeclare was informed by Moody's Physics of Notations (PoN) principles, with insights from expert interviews guiding our approach to refining the notation. However, fully adhering to PoN often involves managing intrinsic trade-offs [56]. For example, accentuating *Graphic Economy* (i.e., reducing the total number of graphical symbols) may limit *Semiotic Clarity*, where a one-to-one mapping between symbols and semantic constructs becomes more challenging to maintain. In developing easyDeclare, we balanced these trade-offs to achieve an effective visual notation. Furthermore, we interviewed experts who analyzed in detail the adherence of Declare, the Hanser et al. notation, and easyDeclare to Moody's principle.

Since the PoN principles generally provide guidelines rather than rigid prescriptions, we also conducted a controlled experiment to evaluate our proposed notation. A further step would be an expert study that analyzes in detail the adherence of the new notation to Moody's principles and a further experiment with practitioners to test the effectiveness and efficacy of the proposed notation.



Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	TOTAL	GRADE
P1	7.5	7.5	7.5	7.5	10.0	7.5	7.5	7.5	10.0	7.5	80.0	A-
P2	5.0	10.0	7.5	7.5	7.5	10.0	5.0	7.5	5.0	10.0	75.0	B
P3	7.5	7.5	7.5	7.5	7.5	7.5	2.5	10.0	5.0	7.5	70.0	C
P4	10.0	10.0	10.0	7.5	7.5	10.0	10.0	10.0	10.0	10.0	95.0	A+
P5	7.5	10.0	10.0	10.0	7.5	10.0	10.0	10.0	10.0	10.0	95.0	A+
P6	5.0	10.0	10.0	7.5	10.0	10.0	10.0	10.0	10.0	10.0	92.5	A+
P7	10.0	10.0	7.5	10.0	7.5	10.0	7.5	10.0	10.0	7.5	90.0	A+
P8	7.5	10.0	10.0	10.0	7.5	10.0	7.5	10.0	10.0	7.5	90.0	A+
P9	5.0	7.5	10.0	7.5	7.5	10.0	10.0	7.5	7.5	10.0	82.5	A
P10	5.0	10.0	5.0	10.0	5.0	10.0	7.5	10.0	10.0	10.0	82.5	A
P11	7.5	7.5	10.0	7.5	10.0	10.0	10.0	7.5	7.5	7.5	85.0	A+
P12	7.5	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	97.5	A+
P13	2.5	10.0	7.5	10.0	7.5	10.0	10.0	10.0	7.5	10.0	85.0	A+
P14	10.0	10.0	10.0	10.0	10.0	7.5	10.0	10.0	10.0	10.0	97.5	A+
P15	7.5	10.0	10.0	7.5	10.0	10.0	10.0	7.5	10.0	10.0	92.5	A+
P16	7.5	10.0	7.5	10.0	10.0	10.0	7.5	10.0	7.5	10.0	90.0	A+
P17	7.5	10.0	10.0	10.0	10.0	7.5	10.0	7.5	10.0	7.5	90.0	A+
P18	7.5	7.5	10.0	10.0	10.0	10.0	7.5	7.5	7.5	10.0	87.5	A+
										AVERAGE	87.64	A+

Fig. 16. System Usability Scale (SUS) results for EDD. These results indicate a high usability level, with an average rating of A+ (score 87.64).

The experiment evaluated the proposed graphical notation against the pre-existing standard, Declare, focusing on performance and perception metrics. In decoding tasks, easyDeclare showed higher effectiveness than Declare, although it took more time, indicating better accuracy but reduced efficiency. For encoding tasks, there were no significant differences in effectiveness or efficiency between the two notations. There was no significant difference in the initial time spent learning both notations. However, easyDeclare required significantly less time for subsequent explanations. In template encoding tasks, easyDeclare showed better performance after initial explanations, though differences became insignificant after further explanations. In both decoding and encoding tasks for models, easyDeclare outperformed Declare in terms of effectiveness and efficiency, suggesting that it handles complexity better. easyDeclare was perceived as easier to use (PEOU) and users were more likely to use it in the future (ITU). There were no significant differences in perceived usefulness (PU) between the two notations.

However, fully adhering to PoN involves managing intrinsic trade-offs. For example, in striving for Graphic Economy, easyDeclare chose to encode template variations (Alternate/Chain) using embedded letters (A/C). The PoN evaluation noted this choice partially impacts Semantic Transparency and Dual Coding.

As a venue for future work, informed by the systematic comparison, we intend to explore how design elements from other notations that satisfy graphic economy, such as the minimalist visual set used by Hanser et al. [6], could be adapted to address these trade-offs in easyDeclare. Specifically, future research will focus on integrating alternative graphical approaches for encoding template variants (A/C) to enhance Semantic Transparency and Dual Coding without sacrificing Graphic Economy, drawing lessons from the range of design choices identified in the multi-notation PoN review. Finally, we aim to explore the understandability of the single Declare and easyDeclare constraints, investigating how specific constraints impact cognitive load and the learning curve for users.

In conclusion, easyDeclare demonstrated superior effectiveness and efficiency in complex tasks and higher user satisfaction in ease of use and intention to use, making it a promising alternative to the traditional graphical notation provided by Declare for declarative process modeling.

Furthermore, we aim to integrate easyDeclare into a declarative process mining suite, such as *RuM* [49], to broaden its usage among practitioners and conduct further longitudinal user tests.

The artifacts presented in this paper are available in the official *easyDeclare* GitHub organization at https://github.com/easyDeclare. Both

the template images for easyDeclare and the open-source code of the EDD tool are available. This organization will be used to maintain and expand the notation, support tool development, and facilitate reproducibility.

CRediT authorship contribution statement

Graziano Blasilli: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Conceptualization. **Lauren S. Ferro:** Writing – original draft, Visualization, Validation, Conceptualization. **Simone Lenti:** Writing – review & editing, Writing – original draft, Visualization, Validation, Investigation, Formal analysis, Conceptualization. **Fabrizio Maria Maggi:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Andrea Marrella:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Tiziana Catarci:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Sapienza project FOND-AIBPM, the MUR PRIN 2022 project "Motown: Smart Production Planning and Control for Manufacturing of Electric Vehicle Powertrain in Industry 4.0 Environment", the MUR PRIN 2022 Project "Discount quality for responsible data science: Human-in-the-Loop for quality data", and the PNRR MUR project PE0000013-FAIR.

Data availability

Data will be made available on request.

References

[1] S. Miksch, C. Di Ciccio, P. Soffer, B. Weber, Visual analytics meets process mining: Challenges and opportunities, IEEE Comput. Graph. Appl. 44 (6) (2024) 132–141.
[2] A. Alman, A. Arleo, I. Beerepoot, A. Burattin, C. Di Ciccio, M. Resinas, Tiramisù: Making sense of multi-faceted process information through time and space, J. Intell. Inf. Syst. (2024) 1–23.
[3] W.M.P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Comput. Sci.-Res. Dev. 23 (2) (2009) 99–113.
[4] H.A. Lopez A., V. Simon, How to (re)design declarative process notations? A view from the lens of cognitive effectiveness frameworks, in: 15th IFIP Working Conference on the Practice of Enterprise Modeling 2022 (PoEM-Forum 2022), 2022.
[5] C. Di Ciccio, T. Catarci, M. Mecella, Representing and visualizing mined artful processes in MailOfMine, in: HCI-KDD, Springer, 2011, pp. 83–94.
[6] M. Hanser, C. Di Ciccio, J. Mendling, A new notational framework for declarative process modeling, Softw.tech. - Trends 36 (2) (2016).
[7] D. Fahland, J. Mendling, H.A. Reijers, B. Weber, M. Weidlich, S. Zugal, Declarative versus imperative process modeling languages: The issue of maintainability, in: Business Process Management Workshops, 2010, pp. 477–488.
[8] C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, B. Weber, Understanding declare models: Strategies, pitfalls, empirical results, Softw. Syst. Model. 15 (2) (2016) 325–352.
[9] K. Figl, C. Di Ciccio, H.A. Reijers, Do declarative process models help to reduce cognitive biases related to business rules? in: 39th International Conference on Conceptual Modeling (ER 2020), Springer, 2020, pp. 119–133.
[10] D.M.T. Trinh, A. Abbad-Andaloussi, H.A. López, On the semantic transparency of declarative process models: the case of constraints, in: Proceedings of the Research Track At International Conference on Cooperative Information Systems 2023, 2023, pp. 217–236.
[11] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, H.A. Reijers, Imperative versus declarative process modeling languages: An empirical investigation, in: Business Process Management Workshops (1), 2011, pp. 383–394.
[12] L.S. Ferro, A. Marrella, VERTO: A visual notation for declarative process models, in: 2018 Int. Conf. on Advanced Visual Interfaces, AVI, ACM, 2018, p. 62.
[13] D. Moody, The "Physics" of notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Trans. Softw. Eng. 35 (6) (2009) 756–779.
[14] G. De Giacomo, M.Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: IJCAI 2013, 23rd International Joint Conference on Artificial Intelligence, 2013.
[15] W.M.P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.
[16] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer, 2013.
[17] F.M. Maggi, R.P.J.C. Bose, W.M.P. van der Aalst, Efficient discovery of understandable declarative process models from event logs, in: 24th Int. Conf. on Advanced Information Systems Engineering (CAiSE), 2012, pp. 270–285.
[18] C. Di Ciccio, M. Mecella, On the discovery of declarative control flows for artful processes, ACM Trans. Manag. Inf. Syst. 5 (4) (2015) 24:1–24:37.
[19] S. Schönig, A. Rogge-Solti, C. Cabanillas, S. Jablonski, J. Mendling, Efficient and customisable declarative process mining with SQL, in: 28th Int. Conf. on Advanced Information Systems Engineering (CAiSE), 2016, pp. 290–305.
[20] S.K.L.M. vanden Broucke, J. Vanthienen, B. Baesens, Declarative process discovery with evolutionary computing, in: 2014 IEEE Congress on Evolutionary Computation, CEC, 2014, pp. 2412–2419.
[21] S. Agostinelli, F. Chiariello, F.M. Maggi, A. Marrella, F. Patrizi, Process mining meets model learning: Discovering deterministic finite state automata from event logs for business process analysis, Inf. Syst. 114 (2023).
[22] F.W. Vaandrager, Model learning, Comm. ACM 60 (2) (2017) 86–95.
[23] T.R. Green, A.E. Blandford, L. Church, C.R. Roast, S. Clarke, Cognitive dimensions: Achievements, new directions, and open questions, J. Vis. Lang. Comput. 17 (4) (2006) 328–365.
[24] J. Krogstie, SEQUAL specialized for business process models, in: Quality in Business Process Modeling, Springer, 2016, pp. 103–138.
[25] D. Van Der Linden, A. Zamansky, I. Hadar, A framework for improving the verifiability of visual notation design grounded in the physics of notations, in: 2017 IEEE 25th International Requirements Engineering Conference, RE, IEEE, 2017, pp. 41–50.
[26] G. Popescu, A. Wegmann, Using the physics of notations theory to evaluate the visual notation of seam, in: 2014 IEEE 16th Conf. on Business Informatics, CBI, vol. 2, 2014, pp. 166–173.
[27] V. Diamantopoulou, H. Mouratidis, Applying the physics of notation to the evaluation of a security and privacy requirements engineering methodology, Inf. Comput. Secur. 26 (4) (2018).
[28] N. Genon, P. Heymans, D. Amyot, Analysing the cognitive effectiveness of the BPMN 2.0 visual notation, in: Int. Conf. on Software Language Eng., SLE, 2010, pp. 377–396.
[29] M.d.G. da Silva Teixeira, G.K. Quirino, F. Gailly, R. de Almeida Falbo, G. Guizzardi, M.P. Barcellos, PoN-S: A systematic approach for applying the physics of notation (PoN), in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2016, pp. 432–447.
[30] C. Boyce, P. Neale, Conducting In-Depth Interviews: A Guide for Designing and Conducting In-Depth Interviews for Evaluation Input, Technical report, Pathfinder International, 2006.
[31] D. van der Linden, I. Hadar, A systematic literature review of applications of the physics of notations, IEEE Trans. Softw. Eng. 45 (8) (2019) 736–759.
[32] C.D. Wickens, C.M. Carswell, The proximity compatibility principle: Its psychological foundation and relevance to display design, Hum. Factors 37 (3) (1995) 473–494.
[33] D. Moody, The method evaluation model: A theoretical model for validating information systems design methods, in: 11th European Conf. Information Systems, ECIS, 2003.
[34] A. Gemino, Y. Wand, Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties, Data Knowl. Eng. 55 (3) (2005) 301–326.
[35] F. Bodart, A. Patel, M. Sim, R. Weber, Should optional properties be used in conceptual modelling? A theory and three empirical tests, Inf. Syst. Res. 12 (4) (2001) 384–405.
[36] H. Reijers, J. Mendling, R. Dijkman, Human and automatic modularizations of process models to enhance their comprehension, Inf. Syst. 36 (5) (2011) 881–897.
[37] W. Wang, T. Chen, M. Indulska, S. Sadiq, B. Weber, Business process and rule integration approaches — an empirical analysis of model understanding, Inf. Syst. 104 (2022) 101901.
[38] A. Abbad-Andaloussi, A. Burattin, T. Slaats, E. Kindler, B. Weber, Complexity in declarative process models: Metrics and multi-modal assessment of cognitive load, Expert Syst. Appl. 233 (2023).
[39] S. Abrahão, E. Insfran, J.A. Carsí, M. Genero, Evaluating requirements modeling methods based on user perceptions: A family of experiments, Inform. Sci. 181 (16) (2011) 3356–3378.

[40] A. Jalali, Evaluating user acceptance of knowledge-intensive business process modeling languages, Softw. Syst. Model. 22 (6) (2023) 1803–1826.
[41] V.R. Basili, G. Caldiera, H.D. Rombach, The goal question metric approach, in: Encyclopedia of Software Engineering, Wiley, 1994, pp. 646–661.
[42] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Trans. Softw. Eng. 28 (8) (2002) 721–734.
[43] M. Höst, B. Regnell, C. Wohlin, Using students as subjects—A comparative study of students and professionals in lead-time impact assessment, Empir. Softw. Eng. 5 (3) (2000) 201–214.
[44] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, Berlin, Heidelberg, 2024.
[45] P.W. Vogt, Dictionary of Statistics & Methodology, SAGE Publications, Inc., 2005.
[46] S. Nagel, P. Delfmann, Exploring cognitive effects of inconsistency characteristics on understanding inconsistencies in declarative process models, in: Proceedings of the 57th Hawaii International Conference on System Sciences, ScholarSpace, Waikiki, United States, 2024.
[47] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, M. Oivo, Empirical software engineering experts on the use of students and professionals in experiments, Empir. Softw. Eng. 23 (1) (2018) 452–489.
[48] V. Skydanienko, C. Di Francescomarino, C. Ghidini, F.M. Maggi, A tool for generating event logs from multi-perspective declare models, in: Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018, 2018.
[49] A. Alman, C.D. Ciccio, D. Haas, F.M. Maggi, A. Nolte, Rule mining with RuM, in: 2020 2nd Int. Conf. on Process Mining, ICPM, 2020, pp. 121–128.
[50] C. Gunther, H. Verbeek, XES - Standard Definition, in: BPM reports, BPMcenter.org, 2014.
[51] I. Donadello, F. Riva, F.M. Maggi, A. Shikhizada, Declare4py: A python library for declarative process mining, in: BPM (PhD/Demos), in: CEUR Workshop Proceedings, vol. 3216, CEUR-WS.org, 2022, pp. 117–121.
[52] J. Brooke, SUS: A quick and dirty usability scale, Usability Eval. Ind. 189 (3) (1996) 189–194.
[53] J. Brooke, SUS: A retrospective, J. Usability Stud. 8 (2) (2013) 29–40.
[54] J.R. Lewis, J. Sauro, Item benchmarks for the system usability scale, J. Usability Stud. 13 (3) (2018).
[55] M.L. Bernardi, A. Casciani, M. Cimitile, A. Marrella, Conversing with business process-aware large language models: The BPLLM framework, J. Intell. Inf. Syst. 62 (6) (2024) 1607–1629.
[56] J. Ziehmann, B. Lantow, Moody's physics of notations: High impact, little support, in: PoEM Forum - the Practice of Enterprise Modeling, 2021, pp. 31–38.