Softw Syst Model
DOI 10.1007/s10270-014-0435-z

SPECIAL SECTION PAPER

Understanding Declare models: strategies, pitfalls, empirical results

Cornelia Haisjackl · Irene Barba · Stefan Zugal · Pnina Soffer · Irit Hadar · Manfred Reichert · Jakob Pinggera · Barbara Weber

Received: 11 October 2013 / Revised: 18 July 2014 / Accepted: 27 August 2014

Abstract Declarative approaches to business process modeling are regarded as well suited for highly volatile environments, as they enable a high degree of flexibility. However, problems in understanding and maintaining declarative process models often impede their adoption. Likewise, little research has been conducted into the understanding of declarative process models. This paper takes a first step toward addressing this fundamental question and reports on an empirical investigation consisting of an exploratory study and a follow-up study focusing on the system analysts' sense-making of declarative process models that are specified in Declare. For this purpose, we distributed real-world Declare models to the participating subjects and asked them to describe the illustrated process and to perform a series of sense-making tasks. The results of our studies indicate that two main strategies for reading Declare models exist: either considering the execution order of the activities in the process model, or orienting by the layout of the process model. In addition, the results indicate that single constraints can be handled well by most subjects, while combinations of constraints pose significant challenges. Moreover, the study revealed that aspects that are similar in both imperative and declarative process modeling languages at a graphical level, while having different semantics, cause considerable troubles. This research not only helps guiding the future development of tools for supporting system analysts, but also gives advice on the design of declarative process modeling notations and points out typical pitfalls to teachers and educators of future systems analysts.

Keywords Declarative process models · Empirical research · Understandability

Communicated by Dr. Selmin Nurcan.

This research is supported by Austrian Science Fund (FWF): P23699-N23 and the BIT fellowship program, by Spanish Ministerio de Ciencia e Innovación (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

C. Haisjackl (✉) · S. Zugal · J. Pinggera · B. Weber
University of Innsbruck, Innsbruck, Austria
e-mail: cornelia.haisjackl@uibk.ac.at

S. Zugal
e-mail: stefan.zugal@uibk.ac.at

J. Pinggera
e-mail: jakob.pinggera@uibk.ac.at

B. Weber
e-mail: barbara.weber@uibk.ac.at

I. Barba
University of Seville, Seville, Spain
e-mail: irenebr@us.es

P. Soffer · I. Hadar
University of Haifa, Haifa, Israel
e-mail: spnina@is.haifa.ac.il

I. Hadar
e-mail: hadari@is.haifa.ac.il

M. Reichert
University of Ulm, Ulm, Germany
e-mail: manfred.reichert@uni-ulm.de

1 Introduction

In the context of analyzing and designing information systems, the positive influence of conceptual modeling on understanding and communication has been documented [42]. For example, *business process models* (*process models* for short) have been employed in the context of process-aware information systems, service-oriented architectures, and Web ser-

Published online: 28 September 2014

vices [54]. Recently, *declarative approaches*, and specifically Declare [47], have received increasing attention due to their flexibility with respect to modeling and execution of processes [52]. While imperative process models specify exactly *how* things must be done, declarative models focus on the logic that governs the interplay of process actions by describing activities that may be performed as well as constraints prohibiting undesired behavior.

Problem statement Existing research has addressed technical issues of declarative process models, such as formalization of semantics [24], maintainability [63], verification [47], and execution [5]. Understandability concerns of declarative models (i.e., the ability to correctly explain the model, and solve problems related to the model), on the contrary, have only been considered to a limited extent. In particular, it has been argued that understandability may be hampered by the lack of computational offloading [66] and the existence of hidden dependencies [67]. Put differently, it remains unclear whether the full potential of declarative modeling can be exploited or whether understandability issues impede upon their successful adoption.

Contribution In this paper, we approach these issues by investigating the sense-making of declarative process models specified in Declare in two studies: an exploratory study focusing on the comprehension of Declare models, and a follow-up study designed to confirm and extend the findings of the exploratory study. In both studies, subjects were asked to voice their thoughts while performing the tasks, i.e., we applied *think-aloud* [16] to gain insights into the reasoning processes. The results of the exploratory study suggest an iterative and sequential way of reading Declare models. This is surprising, as Declare models are designed to convey circumstantial information. Further, the exploratory study identified challenges when system analysts were required to combine several constraints, i.e., when *hidden dependencies* between constraints existed. Additionally, the exploratory study investigated how hierarchy in declarative process models affects their understandability. Interestingly, we did not observe differences regarding the strategies applied by subjects when confronted with hierarchical process models.

In the follow-up study, we aim at confirming and extending the findings of the exploratory study by investigating the observed difficulties regarding hidden dependencies, the combination of constraints, and existence constraints. In addition, we examine strategies for understanding declarative models. Since the sense-making of hierarchical Declare models was already addressed in [68] and no additional strategies for understanding could be observed in hierarchical models, we did not pursue the sense-making of hierarchical models in the follow-up study. In the follow-up study, iterative and sequential reading of Declare models could not be confirmed as a single prevailing strategy for making sense of declarative models, but two distinct strategies could be identified (that coincided in the exploratory study due to the used material). Further, the follow-up study confirmed the challenges observed in the exploratory study like the combination of constraints, hidden dependencies, and pairs of constraints. Existence constraints, in turn, caused relatively little difficulties when asked in isolation. In addition, our study showed that process modeling knowledge on imperative process modeling languages cannot simply be transferred to declarative process modeling. Moreover, the follow-up study revealed that aspects that appear to be similar in imperative and declarative process modeling languages at a graphical level, while having different meaning, caused considerable difficulties.

The exploratory study is part of a larger investigation on declarative process models [68]. While [68] focused on *quantitative* results, the exploratory study contained in this paper and presented in [22] describes solely *qualitative* data. This paper extends the work from [22] with a follow-up study based on the findings of the exploratory study. Therefore, the contribution of this paper is twofold. First, we intend to verify the findings of the exploratory study. Second, we strive for deepening our knowledge regarding the understanding of declarative process models by covering additional research questions that were raised in the exploratory study. In this way, this paper constitutes another building block toward a more comprehensive understanding of the declarative process modeling paradigm. In the long run, we aim at guiding the development of tools for supporting system analysts, as well as pointing out typical pitfalls to teachers and educators of systems analysts. In addition, our research has implications for the design of declarative process modeling notations.

The remainder of the paper is structured as follows. Section 2 gives background information. The exploratory study is described in Sect. 3, whereas the follow-up study is described in Sect. 4. Related work is presented in Sect. 5, and finally, Sect. 6 concludes the paper.

2 Backgrounds

In this section, we present background information on declarative models in general, traces for declarative models, and hierarchy in the context of declarative models. We discuss potential understandability problems of declarative models, i.e., hidden dependencies. Finally, we present the concept of mental effort as an additional measure for understanding.

2.1 Declarative process models

Declarative approaches have received increasing interest, as they suggest a fundamentally different way of modeling busi-

Table 1 Definition of constraints

Group	Constraint	Definition
Existence	exactly(a,n)	Activity *a* must occur exactly *n* times
	existence(a,n)	*a* must occur at least *n* times
	max(a,n)	*a* must occur at most *n* times
	init(a)	*a* must be the first executed activity in every trace
	last(a)	*a* must be the last executed activity in every trace
Relation	precedence(a,b)	activity *b* must be preceded by activity *a* (not necessarily directly)
	response(a,b)	If *a* is executed, *b* must be executed afterward (not necessarily directly afterward)
	succession(a,b)	Combines precedence(a,b) and response(a,b)
	chain_response(a,b)	If *a* is executed, *b* is executed directly afterward
	coexistence(a,b)	If *a* is executed, *b* must be executed and vice versa
Negation	neg_response(a,b)	If *a* is executed, *b* must not be executed afterward
	neg_coexistence(a,b)	*a* and *b* cannot co-occur in any trace

ness processes [47]. Instead of describing *how* a process must be executed, declarative models focus on the logic that governs the interplay of process actions. For this purpose, declarative process models specify *activities* that may be performed as well as *constraints* prohibiting undesired behavior. Constraints may be divided into existence, relation, and negation constraints [1]. *Existence constraints* specify how often an activity must be executed for one particular process instance. In turn, *relation constraints* restrict the ordering of activities by imposing corresponding restrictions. Finally, *negation constraints* define negative relationships between activities. Table 1 shows examples for each category; an overview of all constraints can be found in [1].

2.2 Traces for declarative process models

A *trace* is defined as a *completed* process instance [52]. It can have two different states. If it satisfies all constraints of the model, it is denoted as *valid* (also referred to as *satisfied*). If the trace violates constraints in the model, it is considered as *invalid* (also referred to as *violated*). A *minimal trace* is defined as a valid trace with a minimum number of activities. Put differently, a trace is minimal if there exists no other valid trace that contains less activities. A *sub-trace*, in turn, can be in three different states:

- *Valid*: The sub-trace satisfies all constraints of the process model.
- *Temporarily violated*: The sub-trace does not satisfy all constraints of the process model, but there is an affix or suffix that could be added to the sub-trace such that all constraints are satisfied.
- *Invalid*: The sub-trace violates constraints in the process model, and no affix or suffix can be added to the sub-trace to satisfy all constraints.

An example of a Declare model S is depicted in Fig. 1 [47].[1] The model consists of six distinct activities A, B, C, D, E, and F. In addition, it comprises three constraints. The *neg_coexistence* constraint (i.e., C1) forbids that A and B co-occur in the same trace. In turn, the *response* constraint (i.e., C2) requires that every execution of C must be followed by one of F before the process instance may complete. Finally, the *exactly* constraint (i.e., C3) states that F must be executed exactly once per process instance. The traces $\sigma_1 = \langle A, A, D, E, A, F \rangle$, $\sigma_2 = \langle B, C, F, E, B \rangle$, and $\sigma_3 = \langle B, E, F \rangle$ satisfy all constraints (C1–C3), i.e., σ_1, σ_2, and σ_3 are valid traces. Traces σ_4 to σ_6 are invalid: $\sigma_4 = \langle A, F, C, E, A \rangle$ violates C2, $\sigma_5 = \langle B, D, F, C, F \rangle$ violates C3, and $\sigma_6 = \langle A, D, B, F, E \rangle$ violates C1. Trace $\sigma_7 = \langle F \rangle$ is the minimal trace since there exists no other valid trace comprising a lower number of activities. Finally, trace $\sigma_8 = \langle D, A, C \rangle$ corresponds to a temporarily violated sub-trace, as C1 and C2 are not satisfied, but the suffix $\langle F \rangle$ could be added such that the sub-trace becomes a valid trace.

2.3 Hierarchy in declarative process models

Using modularization to hierarchically structure information has been identified as a viable approach to deal with complexity for decades [46]. In the context of declarative process models, hierarchy can be established using sub-processes, which are referenced via *complex activities*. When executing such a complex activity, the referred process model, i.e., the sub-process, is instantiated (see [68] for details). Similar to the notion of hierarchy in, e.g., BPMN, the life cycle events of an instance of a complex activity and the respective process instance are coupled. For example, when starting a complex activity, the respective process instance is set

[1] Declare was formerly known as ConDec, see: http://www.win.tue.nl/declare/2011/11/declare-renaming/.

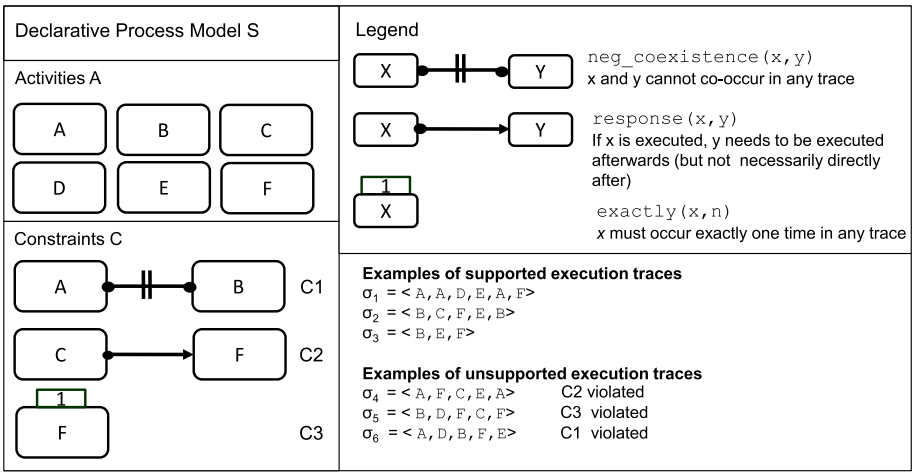


Fig. 1 Example of a declarative process model, adapted from [52]

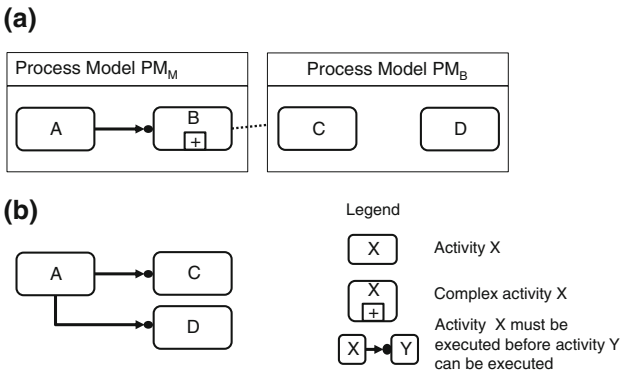


Fig. 2 Example of a process model with and without hierarchy. **a** Hierarchical process model. **b** Corresponding flat process model

to state started. Similarly, the complex activity instance is set to state completed when the referred process instance is completed. Please note that thereby constraints referring to a complex activity do *not directly* influence the activities within the process, the complex activity is referring to. Rather, constraints influence the life cycle of the process instance, which in turn impacts the activities contained therein. To illustrate this notion of hierarchy, consider process model PM_M shown in Fig. 2a. It contains activity A, which is connected by a *precedence constraint* to complex activity B. Complex activity B, in turn, refers to sub-process PM_B containing activities C and D, without any constraints. When translating this model into a flat process model, one might use the following reasoning: In PM_M, no constraint restricts the execution of A; however, the execution of C and D must be preceded by an execution of A (C and D can only be executed within complex activity B, which in turn requires activity A to be executed before). Hence, to describe this behavior without the use of hierarchy, the process model shown in Fig. 2b may be used. Please note that even though Fig. 2a, b are semantically equivalent, they differ in the number of activities and constraints.

2.4 Hidden dependencies

Concerning the understanding of declarative process models, it has been hypothesized that the combination of constraints poses a considerable challenge [39,47,67]; especially, interactions that are not easily recognizable, i.e., *hidden dependencies* [20], are assumed to constitute a significant challenge in reading and thus understanding declarative process models. Consider, for instance, the combination of existence constraints and response constraints as shown in Fig. 1. The combination of the *exactly* constraint (i.e., C3) and the *response* constraint (i.e., C2) adds an implicit constraint that prohibits the execution of `C` after `F`. In particular, C2 requires `C` to be followed by `F`, but at the same time C3 prescribes that `F` must be executed exactly once. In other words, if `C` was executed after `F`, another execution of `F` is required in order to satisfy C2; however, this is prohibited by C3. Since this interaction is not *explicitly* visible, it is not sufficient that the system analyst relies solely on the information displayed explicitly in the process model. Instead, the system analyst must carefully examine the process model for the presence of such hidden dependencies.

2.5 Mental effort

In this work, we investigate the sense-making of declarative process models. Hence, characteristics of the human cognitive system are of interest as well. In the following, we discuss *working memory*, which is essential for effective functioning of the cognitive system. Working memory is responsible for maintaining and manipulating a *limited* amount of information for goal-directed behavior, such as the interpretation of a declarative process model (cf. [4]). As opposed to working memory, *long-term memory* represents a theoretically unlimited information store that contains the complete knowledge base of a person (e.g., knowledge about facts, events, rules, and procedures). Working memory strongly interacts with

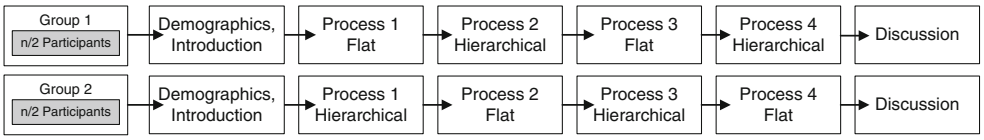


Fig. 3 Design of the exploratory study

long-term memory. For understanding a declarative process model, for instance, knowledge about the modeling notation is necessary. In this context, working memory is the *work place* where information is integrated, manipulated, and related. Therefore, working memory can be conceptualized as the activated part of *long-term memory* [61]. In [37], the capacity of working memory is stated with 7 ± 2 chunks. More recent works have reported an even smaller capacity of 3–5 items [15]. The amount of working memory currently used is thereby referred to as *mental effort* [44]. Research indicates that a high mental effort increases the probability of errors, especially when the working memory capacity is exceeded [60]. In the context of conceptual models, mental effort is of interest as it appears to be connected to performance, e.g., properly answering questions about a model. Similarly, Moody [40] argues that higher mental effort is in general associated with lower understanding of models.

Measuring mental effort Various techniques exist for assessing mental effort, including pupillometry, i.e., measuring the diameter of the eyes' pupils, heart rate variability, and rating scales [44]; especially, rating scales, i.e., self-rating mental effort, have been shown to reliably measure mental effort and is thus widely adopted [44]. Furthermore, this kind of measurement can be easily applied, e.g., by using 7-point rating scales. For instance, in [33], mental effort was assessed using a 7-point rating scale, ranging from *(1) very easy* to *(7) very hard* for the question "How difficult was it for you to learn about lightning from the presentation you just saw?" In the context of conceptual modeling, it was argued that mental effort should be considered as an additional measure of understanding together with accuracy and duration [2,65]. For instance, in contrast to accuracy, subtle differences can presumably be observed [65]. In particular, for cases where mental effort is well within the working memory's limits and thus does not provoke a significant number of errors, still a difference in mental effort can be observed [65].

3 Empirical investigation part 1: exploratory study

Since there has been no considerable research on understandability issues of declarative process models, and hence no theories exist, we can base our investigation on and we address the topic in an exploratory manner using a qualitative research approach [6]. In particular, we use the think-aloud method, i.e., we ask participating subjects to voice their thoughts, allowing for a detailed analysis of their reasoning process [16]. Then, we turn to grounded theory [13], an analysis approach for identifying recurring aspects and grouping them to categories. These categories are validated and refined throughout the analysis process.

3.1 Defining and planning the exploratory study

3.1.1 Research question

Goal of this study was to investigate how system analysts make sense of declarative process models. In particular, we are interested in common strategies and typical pitfalls occurring during this sense-making process. The research question RQ_0 can be stated as follows:
Research question RQ_0 What are common strategies and typical pitfalls that can be observed when system analysts make sense of declarative process models?

3.1.2 Subjects

In order to ensure that obtained results are not influenced by unfamiliarity with declarative process modeling, subjects need to be sufficiently trained. Even though we do not require experts, subjects should have at least a moderate understanding of declarative processes' principles. For information on the actual subjects, see Sect. 3.2.1.

3.1.3 Objects

The models used in the study originate from a case study [21] and describe real-world business processes. From a set of 24 process models collected in this case study, 4 models were chosen as basic *objects* for the exploratory study. This was accomplished in a way ensuring that the numbers of activities and constraints vary. To make the models amenable for this study, they underwent the following procedure. First, the models were translated to English (the case study was conducted in German) since all exercises were done in English. Second, since the models collected during the modeling sessions had not gone through quality assessment, they were scanned for errors and corrected accordingly. Third, since we were interested in how different structures of the process representation would influence the process models' understand-

ability, we created a second variant of each process describing the same process, but making use of sub-processes. Consequently, we have two variants of each process model: a flat and a hierarchical one. The characteristics of the process models P_1 to P_4 are described in [22].

3.1.4 Design

Figure 3 shows the overall design of the exploratory study: first, subjects are *randomly* assigned to two groups of similar size. Regardless of the group assignment, demographical data are collected and subjects obtain introductory assignments. To support subjects in their task, cheat sheets briefly summarizing the constraints' semantics are provided, which can be used throughout the study. Introductory tasks allow subjects to familiarize themselves with the type of tasks to be performed—potential problems can therefore be resolved at this early stage without influencing actual data collection. After this familiarization phase, subjects are confronted with the actual tasks. Each subject works on two flat process models and two hierarchical ones. Group 1 starts with the flat representation of process model 1, while Group 2 works on the hierarchical representation of the same model. Subjects are confronted with hierarchical and flat models in an alternating manner. For each model, the subject is asked to "explain roughly what the process describes." The study is concluded by a discussion with the subject to help reflecting on the study and providing us with feedback.

3.1.5 Instrumentation

For each model, subjects received separate paper sheets showing the models, allowing them to use a pencil for highlighting or taking notes. No written answers were required, only free talking. Audio- and video recording was used as it has proven being useful for resolving unclear situations in think-aloud protocols [63].

3.2 Performing the exploratory study

3.2.1 Execution

The study was conducted in July 2012 in two locations. First, seven subjects participated at the University of Ulm, followed by two additional sessions at the University of Innsbruck, i.e., a total of nine subjects participated. Even though we have a small sample size, we want to mention that—as described in Sect. 3.4—the sample size is not unusual for this kind of empirical investigation due to the substantial effort to be invested per subject (e.g., transcribing video data and qualitative analysis). According to the qualitative theoretical sampling principles [59], our subjects represent an appropriate sample, as all of the subjects were familiar with Declare (cf. Sect. 3.2.2), and differ in other aspects, such as university and location. Additionally, to ensure that subjects were sufficiently familiar with declarative process modeling, they were provided with training material. Each session was organized as follows: first, the subject was welcomed and instructed to speak thoughts out loudly. To allow subjects to concentrate on their tasks, the sessions were performed in a "paper-workflow" manner, i.e., one supervisor was seated left to the subject and a second supervisor to the right. The sheets containing the study's material were then passed from the left to the subject. As soon as the subject finished the task, the material was passed to the supervisor on the right.[2] Meanwhile, the subject's actions were audio- and video recorded to gather any uttered thoughts.

3.2.2 Data validation

In each session, only a single subject participated, allowing us to ensure that the study setup was obeyed. In addition, we screened whether subjects fitted the targeted profile, i.e., whether they were familiar with process modeling in general and Declare [47]. We asked questions regarding familiarity on process modeling, Declare, and domain knowledge; note that the latter might significantly influence performance [29]. We conclude that they had a profound background in process modeling (the least experienced subject had 2.5 years of modeling experience) and were moderately familiar with Declare (see [22] for details). Finally, we assessed the subjects' professional background: all subjects indicated an academic background, i.e., they were either Ph.D. students or postdocs.

3.2.3 Data analysis

Our research focuses on sense-making of declarative process models. On one hand, we investigated strategies applied by subjects in understanding process models, and on the other, we explored typical phenomena and pitfalls in this process. For this purpose, data analysis comprised the following stages.

1. Transcribing the verbal utterances
2. Creating graphs describing the order in which subjects mention activities
3. Analyzing transcripts using grounded theory.

In (2), for each model, we created a graph representing the order activities were mentioned by the subjects. For this purpose, we utilized the transcripts created in (1), but also video recordings to identify when subjects visited an activity without talking about it. In (3), we applied grounded theory to

[2] The exploratory study's material can be downloaded from http://bpm.q-e.at/experiment/HierarchyDeclarative.

the transcripts in order to explore and understand phenomena appearing when subjects make sense of Declare models. As a starting point, transcripts were inspected in order to mark aspects that caused confusion and were misinterpreted or left out. In a second iteration, we revisited the marked areas and searched for new aspects. This process of open coding analysis was repeated until no new aspects could be found. Afterward, we performed axial coding, i.e., we repeatedly grouped aspects to form high-level categories. We counted the number of identified markings per category.

3.3 Findings of the exploratory study

Based on the findings of our data analysis, we identified different ways how declarative models are read and interpreted answering research question RQ_0. For further details and examples, please take a look at [22].

3.3.1 Reading declarative business process models

When analyzing graphs and transcripts, we observed that subjects consistently adopted similar strategies when reading declarative models. Regardless of whether sub-processes were present or not, they described the process in the order activities that were supposedly executed, i.e., they tried to describe the process in a *sequential way*. Hence, as a first step, subjects skimmed over the process model to find an *entry point* where they could start with describing the (main) process. A declarative process model, however, does not necessarily have a unique entry point, apparently causing confusion. The subjects used two different solutions for this kind of situation. Either they looked for a last constraint, or they assumed the upper left corner of the model to be its entry point. After having identified an entry point, subjects tried to figure out in which order activities are to be executed.

This routine was iterative, i.e., if parts of a model were not connected, subjects applied the same strategy for each component, i.e., they started again at the upper left corner of these components. We observed this behavior independent of the respective process model or subject. Finally, subjects indicated where the process supposedly ends. When there was no last constraint, subjects stopped describing the process model after having mentioned all activities of all components. If a model contained sub-processes, subjects preferred talking first about the main process in the above-specified way before describing the sub-processes. When reading sub-processes, the subjects used the same routine as for the main process, except two subjects. One of them described all and the second subject one out of four sub-processes completely *backwards*, i.e., following the semantics of precedence constraints, instead of describing them sequentially.

3.3.2 Single building blocks

Flat declarative process models In general, when subjects made sense of a model, they named activities and their connections. Sometimes, it happened that subjects missed single or small groups of activities. In summary, 27 out of 294 activities were missed in flat process models. When describing a model sequentially, subjects named activities explicitly and most of the connections, i.e., the constraints, implicitly. However, most subjects did not mention existence constraints. This behavior could not be found for any other constraint. For 12 out of 18 models (9 subjects described two flat models), subjects left out 34 of 78 existence constraints in flat models.

Hierarchical declarative process models Regarding hierarchical process models, subjects tended to miss less activities. In summary, 5 out of 331 activities were missed in hierarchical process models. Concerning the existence constraints in hierarchical process models, for 11 out of 18 models (9 subjects described two hierarchical process models), one or more existence constraints were not mentioned. 52 out of 117 existence constraints were ignored in hierarchical process models.

Flat and hierarchical declarative process models As far as the interpretation of constraints is concerned, subjects had relatively little problems irrespective of whether the models were flat or hierarchical. 12 different constraint types were used in the experimental material. To accomplish their task, subjects had cheat sheets available and could look up constraints they did not know. Except for the precedence constraint, which caused considerable difficulties, subjects faced no notable problems. Four out of nine subjects used the precedence constraint in a wrong way. The definition of this constraint (cf. Sect. 2) is that "B may only be executed, if A has been executed before." The subjects used it the other way round, i.e., "So if we perform [A], then [B] should be performed afterward…."

3.3.3 Combination of constraints

Constraints between two activities P_1 contained two and P_4 five situations where two constraints link two activities. In 6 out of these 7 cases, the direction of the constraint arrows is directly opposed to each other. For example, one needs to get offers for interior of an apartment before buying them (precedence constraint). After the interior is bought, it is not reasonable to get new offers (negation response). The subjects had no troubles to understand these situations. However, in the first process model, there is a case where a precedence constraint and a chained response constraint link the two

activities "write test" and "run tests." Both arrows are pointing to the second activity. The precedence constraint ensures that before the first execution of "run tests," "write test" must be executed at least once, i.e., it is not possible to run a test before it was written. The chained response constraint tells us that "If A has been executed, B must be executed immediately afterward," meaning that after the test was written, it must be run directly afterward; 4 out of 9 subjects had troubles with the precedence constraint. Two of them claimed that it is redundant, two even thought it is wrong. The other 5 subjects ignored the precedence constraint.

Hidden dependencies All process models contain hidden dependencies (cf. Sect. 2). Since these interactions are not *explicitly* visible, it is not sufficient that the system analyst only relies on the information displayed explicitly, but must carefully examine the process model for these hidden dependencies as well. Our results show that the subjects mostly ignored hidden dependencies, i.e., only in 8 out of 36 cases (4 models per subject, 9 subjects), a hidden dependency was mentioned or found.

3.4 Exploratory study: discussion

3.4.1 Reading declarative process models

Subjects preferred describing process models in an iterative and sequential way. The sequential way of describing models is surprising, as it is known that declarative process models rather convey circumstantial information (overall conditions that produce an outcome) than sequential information (how the outcome is achieved) [17]. In other words, in an imperative model, sequences are made explicit, e.g., through sequence flows. In a declarative process model, however, such information might not be available at all. As subjects tend to talk about declarative models in a sequential manner, it appears as if they prefer this kind of information. Interestingly, similar observations could be made in a case study on declarative process modeling [63]. Therein, sequential information, such as "A before B" or "then C," was preferred for communication.

3.4.2 Single building blocks

Regarding the interpretation of single building blocks, subjects mentioned activities and constraints when trying to understand the model. Overall, they had relatively little problems with the interpretation of single building blocks. Exceptions seem to be precedence and existence constraints. As a possible explanation, these constraints are too simple and are thus not mentioned at all; further, cheat sheets are not used (cf. dual-process theory [28] describing the interplay of implicit unconscious and explicit controlled processes). Another explanation is that subjects were biased by previous knowledge about imperative models. Regarding the precedence constraint, it nearly looks like the arrow used in imperative process modeling notations.

3.4.3 Combining constraints

The interplay of constraints seems to pose a challenge, especially in the context of hidden dependencies. One explanation could be that subjects simply forgot looking for them, as reading declarative models can quickly become too complex for humans to deal with [47]. As mentioned earlier, in 8 out of 36 cases, subjects found a hidden dependency. In 5 of these 8 cases, they were found in the second process model, which has the smallest number of activities, constraints, and constraint types. This indicates that, if a model is not too complex, subjects will be able to find hidden dependencies. Given this finding, it seems plausible that the *automated* interpretation of constraints can lead to significant improvements regarding the understandability of declarative process models [63].

3.4.4 Differences between flat and hierarchical process models

Subjects did not distinguish between flat and hierarchical process models when reading the models. They used the same description strategy for components and sub-processes. Interestingly, subjects left out more activities in flat than in hierarchical process models (cf. Sect. 3.3.2). A reason for this phenomenon could be *abstraction* [64], i.e., hierarchy allows aggregating model information by hiding the internals of a sub-process using a complex activity, thereby information can be easier perceived. All other aspects we found could be observed in flat and hierarchical models equally. However, it cannot be excluded that for declarative process, models with more complex sub-processes behavior might change.

3.4.5 Limitations

This study has to be viewed in light of several limitations. First, the number of subjects in the study is relatively low (9 subjects), hampering result generalization. Nevertheless, it is noteworthy that the sample size is not unusual for this kind of empirical investigation due to the substantial effort to be invested per subject [14,43]. Second, even though models used in this study vary in the number of activities, number of constraints, and existence of sub-processes, it remains unclear whether results are applicable to declarative process models in general, e.g., more complex models. In addition, it is unclear whether the results would be the same if models with a different layout would be used. Third, all participating subjects indicated academic background, limit-

ing result generalization. However, subjects indicated profound background in business process management. Hence, we argue that they can be interpreted as proxies for professionals. Lastly, note that this study focuses exclusively on Declare models, and further studies are needed to establish whether the same conclusions would apply to other declarative process modeling languages.

4 Empirical investigation part 2: follow-up study

In the exploratory study described in Sect. 3, we have taken a rather broad perspective on the sense-making of Declare models. Particularly, we observed our subjects in an exploratory manner. The goal of the follow-up study described in this section was to take up the findings from the exploratory study for a more systematic and directed investigation of the identified issues. In particular, we will test our assumption that system analysts tend to show a sequential way of reading Declare models. We will deepen our analysis of potential pitfalls, i.e., problems that potentially occur when interpreting these models. We also consider the influence of knowledge on the understanding of declarative models and take subjective factors into account by asking system analysts for improvements of declarative modeling.

4.1 Defining and planning the follow-up study

In this study, we deepen our investigation by shifting the focus in two ways. *First*, we have found that system analysts were able to make use of modularization in Declare models, i.e., properly interpreted sub-processes. At the same time, we have observed that system analysts tend to apply different reading strategies when reading a Declare model. Therefore, we have shifted our focus toward the investigation of reading strategies and did not make use of modularization anymore, but applied different layout variants to Declare models. *Second*, since this study is rather of confirmatory than of exploratory nature, we also take quantitative data into account by asking research questions regarding the pitfalls we found in the exploratory study. In the following, we introduce the research questions and describe subjects, objects, design, and instrumentation of the follow-up study.

4.1.1 Research questions

The research questions are directly derived from the findings of the exploratory study. In particular, research question RQ_1 investigates how system analysts read declarative process models. In turn, research questions $RQ_{2.1}$ to $RQ_{2.4}$ examine challenges and difficulties system analysts face when interpreting declarative process models. Finally, research questions $RQ_{3.1}$ to $RQ_{3.3}$ focus on measures for improving the understanding of declarative process models. Regarding research question RQ_1, in the exploratory study, we observed that process models were described in the order in which activities are supposedly executed. However, in the exploratory study, we laid out models from top left to bottom right, and it is known that the top left is a common starting place for problem-solving tasks [19]. Thus, we could not exclude that this behavior had been triggered by the layout of the process models. Therefore, in research questions RQ_1, we want to clarify whether these reading strategies are indeed inherent to declarative process models or rather caused by specific layouts.

Research question RQ_1 Which strategies are adopted by system analysts when making sense of declarative process models?

In the exploratory study, we have analyzed the way in which system analysts make sense of declarative process models in a rather broad manner, i.e., how system analysts describe the process model. As it is known that the interpretation of constraints can cause considerable difficulties (cf. [47,67]), we extend our investigation by taking into account situations where several constraints need to be combined. In particular, we investigate challenges that occur when performing basic interpretation tasks, such as the naming of minimal traces, valid traces, or invalid traces (cf. Sect. 2.2):

Research question $RQ_{2.1}$ What are the challenges system analysts are facing when performing basic model interpretation tasks, such as determining minimal traces, valid traces, and invalid traces?

In the exploratory study, we observed potential pitfalls of declarative process models caused whenever two activities were connected by a pair of constraints. In the follow-up study, we investigate this previously identified issue in more detail, as postulated in the following research question:

Research question $RQ_{2.2}$ What are the challenges that arise when system analysts have to deal with pairs of constraints between two activities?

In the exploratory study, we found that system analysts rarely mentioned hidden dependencies. However, it is unclear whether system analysts refrained from mentioning them because they consider them to be trivial or whether they were unaware of them. As it has been claimed in several theoretical works that hidden dependencies negatively influence the understandability of declarative process models [39,67], it appears likely that system analysts were not aware of the hidden dependencies. Thus, we deepen the investigation of hidden dependencies in research question $RQ_{2.3}$ as follows:

Research question $RQ_{2.3}$ What are the challenges that arise when system analysts have to deal with hidden dependencies?

Finally, we have found that existence constraints were largely neglected. In Sect. 3.4, we have speculated that this

could be traced back to the fact that existence constraints are particularly easy to understand and are therefore not explicitly mentioned. The goal of research question $RQ_{2.4}$ was to investigate this claim.

Research question $RQ_{2.4}$ Do challenges arise when system analysts have to deal with existence constraints? If yes, how do system analysts deal with these challenges?

So far, we have focused on aspects that might negatively influence the understanding of declarative models. In the following, we turn toward a more positive perspective and look into factors connected to a better understanding of declarative models. In particular, it has been shown that for *imperative* models, education and experience play a central role regarding understanding [54]. In this vein, in research question $RQ_{3.1}$, we investigate whether similar effects can be observed for declarative models.

Research question $RQ_{3.1}$ Are modeling experience and education connected to a lower error rate in interpreting declarative process models?

With respect to improving understandability of declarative process models, the interpretation of constraints appears to play a central role [47,67]. In this sense, the concept of *mental effort* was associated with lower error rates [65]. To investigate this claim, the link between mental effort and correct answers is examined in research question $RQ_{3.2}$:

Research question $RQ_{3.2}$ Is there a relationship between the number of correctly answered questions and the mental effort spent?

Up to now, research questions focusing on improving understandability were motivated by previous research. In research question $RQ_{3.3}$, we take a broader perspective and ask for particularly challenging modeling constructs as well as personal suggestions for improving the understandability of declarative process models.

Research question $RQ_{3.3}$ Which modeling constructs are perceived particularly challenging and where do system analysts see a potential for improving the understandability of declarative process models?

4.1.2 Subjects

As in the exploratory study, we require sufficiently trained subjects. Again, even though we do not require experts, subjects should have at least a moderate understanding of the principles of declarative processes. For information on the actual subjects, see Sect. 4.2.1.

4.1.3 Objects

From the set of 4 models used in the exploratory study, 2 flat versions were chosen as basic objects for the follow-up study, i.e., P_1 and P_3. The models were adapted to the needs of this study, resulting in P_5 and P_6. The models from the exploratory study were adapted as detailed in the following. First, constraints and components were added or changed to make the models amenable for this study (e.g., as P_3 did not contain a pair of constraints, we added one in P_6). Second, since we were interested in the influence of differences regarding the models' layout on the models' understandability, we created a second variant of each model describing the exact same process, but with a horizontally and vertically mirrored layout. The two variants for P_5 are illustrated in Fig. 4. Consequently, we have two variants of each model: a normal and a mirrored one.

The models vary regarding the number of activities (between 12 and 24 activities), number of constraints (between 18 and 25 constraints), and degree of interconnectivity of constraints, i.e., models consist of three to six components. The models are based on two different domains describing bug fixing in a software company and a worker's duties at an electronic company. Similar to the exploratory study, the process models contain constraints of all three types, i.e., existence, relation, and negation constraints.[3]

4.1.4 Design

The study is designed to prevent potential influencing effects. Specifically, we try to avoid any effects caused by the specific process models, as well any learning or fatigue effects, which could lead to biased results. For this, we alternate the order of models and the different layout variants, as illustrated in Fig. 5a. During the study, each subject receives one normal and one mirrored model. Further, we divide the subjects into four groups; each group receives one of the four possible combinations of process models, layouts, and the order in which they are presented to the subjects. The study is concluded by a discussion with the subject to help reflecting on the study and providing us with feedback. For each process model, a series of questions is asked (cf. Fig. 5b) as detailed subsequently.

Describe process model First, the subjects are asked to describe the process model roughly. This allows the familiarization with the process model. The graph analysis is based on the answers to this question.

Specific questions We ask the subjects three questions regarding traces in declarative process models: name a minimal trace, two valid traces, and two invalid traces. The answers to this question provide an immediate indication whether the subject understood declarative process modeling. Further, a series of questions is designed based on the

[3] The follow-up study's material can be downloaded from: http://bpm.q-e.at/experiment/MakingSenseDeclarative.

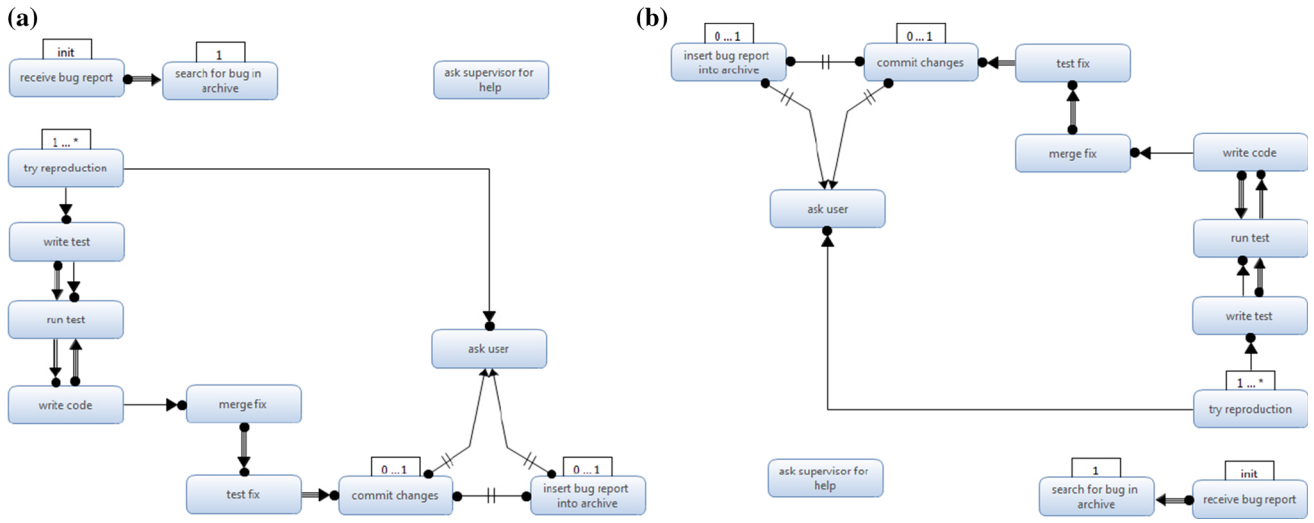


Fig. 4 Normal and mirrored version of P_5. **a** P_{5_normal}, **b** $P_{5\_mirrored}$

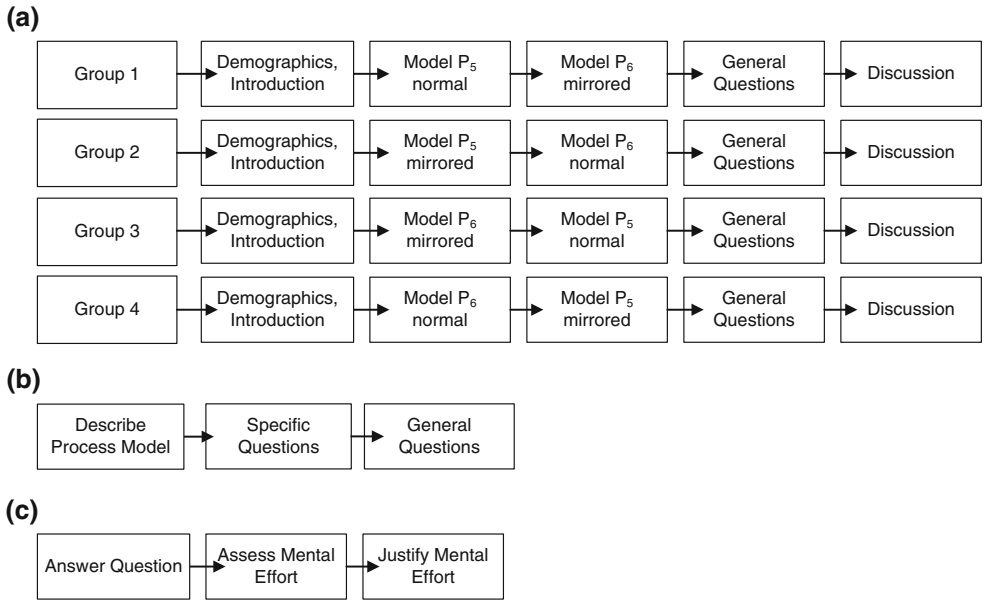


Fig. 5 Design of the follow-up study. **a** Overview, **b** questions per model, **c** tasks per question

findings of the exploratory study to investigate hidden dependencies, pairs of constraints, and combinations of constraints. To limit the influence of "badly formulated" questions, we ask two questions for each category. Additionally, two question on existence constraints are asked since existence constraints were mostly neglected during the exploratory study. In order to gain a more detailed understanding of hidden dependencies, we directly ask the subjects whether they identified any hidden dependencies, and ask them to name examples for hidden dependencies in the process model.

General questions We ask the subjects whether the model is difficult to understand, what part is most challenging, and if they have any suggestions to make the model easier to read/understand.

For each of the questions, in turn, a three-step procedure is followed, cf. Fig. 5c. First, the subject is asked to answer the question. All questions could be answered by using only information provided in the process models, i.e., the questions are *schema-based* comprehension tasks [29]. Likewise, the process models are made available to the subjects while answering the question, i.e., the tasks can be considered to be *read-to-do* (cf. [9]). We use only closed questions, i.e., each question can be answered by selecting from the following answers: *True*, *False*, and *Do not Know*. We award one point for each correct answer and zero points for a wrong

answer (including *Do not Know*). We deliberately allow for the option *Don't Know*, as otherwise subjects would be forced to guess. Second, the subject is asked to assess the expended mental effort. Third, the subject is asked to explain why certain mental effort was indicated.

4.1.5 Instrumentation

Like in the exploratory study, subjects receive separate paper sheets showing the process models for each model, allowing them to use a pencil for highlighting or taking notes as well as juxtaposing the process models as desired. Again, audio- and video recording are used.

4.2 Performing the follow-up study

4.2.1 Execution

The study was conducted in June and July 2013 at three locations. First, one subject participated at the Universitat Politècnica de València, followed by six session at the University of Seville and eleven sessions at the University of Ulm, i.e., a total of 18 subjects participated. Again, we want to mention that a small sample size is not unusual for this kind of empirical investigation (cf., Sect. 4.4). Section 4.2.2 shows that the subjects represent an adequate sample as all of the subjects were familiar with Declare, and differ in other aspects, such as university and location (cf. [59]). Similar to the exploratory study, the subjects were provided with training material. The follow-up study was organized the same way as the exploratory one, with the difference in having only one supervisor in the room handling the sessions in a "paper-workflow" manner. To ensure that the think-aloud sessions were executed in the same way at all places, supervisors were trained in face-to-face meetings and got detailed instructions.

4.2.2 Data validation

We screened subjects for prior knowledge on declarative process modeling. Therefore, we had to omit two subjects due to unfamiliarity with declarative process modeling. Unfortunately, this resulted in unbalanced sizes for the four groups (cf. Fig. 5). Group 1 consisted of 5 subjects, 4 subjects remained in group 2, group 3 contained 4 subjects, and 3 subjects were assigned to group 4. The subjects from the follow-up study all indicated an academic background were either Ph.D. students, postdocs, or professors. We conclude that they had a profound background in process modeling (the least experienced subject had 3 years of modeling experience) and were moderately familiar with Declare. In contrary to the exploratory study, we differentiated the questions regarding process modeling by imperative and declarative process modeling. Additionally, we asked questions with respect to the subjects' knowledge of modeling languages: all subjects started learning process modeling with an imperative modeling language. Only three subjects used a declarative modeling language as their main process modeling language. Demographics are presented in Table 2.

Table 2 Demographics of the follow-up study (8–12 based on 7-point Likert scale)

	Min	Max	Median
(1) Years of modeling experience	3	15	4.5
(2) Imperative models read last year	10	300	50
(3) Imperative models created last year	2	150	15
(4) Average number of activities	8	30	15
(5) Declarative models read last year	0	220	6
(6) Declarative models created last year	0	50	2
(7) Average number of activities	5	15	10
(8) Familiarity Declare	2	7	3.5
(9) Confidence understanding Declare	2	7	3.5
(10) Confidence creating Declare	2	7	4
(11) Familiarity software development	1	7	3.5
(12) Familiarity electronic companies	3	7	5.5

4.3 Findings of the follow-up study

In the previous section, we have discussed the design and execution of the follow-up study. In the following, we use the gathered data to investigate research questions RQ_1 to $RQ_{3.3}$.

4.3.1 RQ_1: Which strategies are adopted by system analysts when making sense of declarative process models?

To target this research question, we applied the same graph analysis procedure based on think-aloud protocols as in the exploratory study (cf., Sect. 3.2).

Normal declarative process models By analyzing graphs and transcripts, we observed that subjects tended to read declarative process models with a normal layout as described in Sect. 3.3.1, i.e., we could replicate the findings of the exploratory study. For example, Fig. 6 shows P_{5_normal} and a typical strategy to understand that model. The model consists of three components. The first one contains activities "receive bug report" and "search for bug in archive." The second component consists of the single activity "ask supervisor for help." The third component, in turn, comprises all remaining activities. The dotted arrows display a typical way in which subjects read the model to understand it.

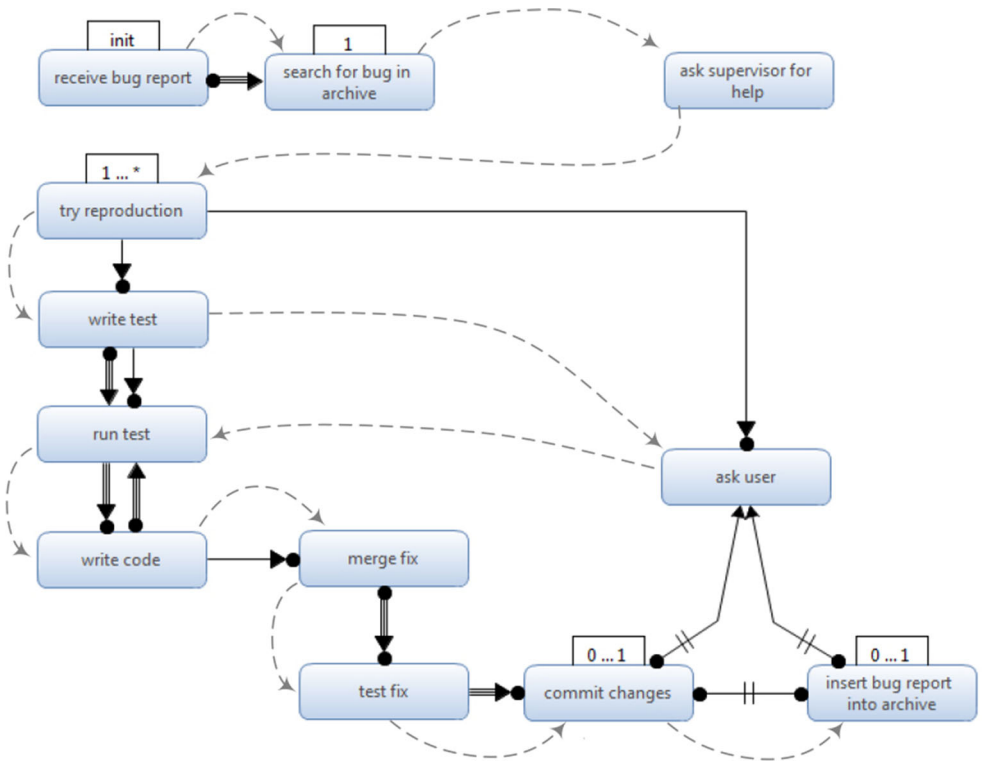


Fig. 6 P_{5_normal} and a reading variant

As a first step, like in the exploratory study, subjects skimmed over the process model to find an *entry point* where they could start with describing the process: "The process starts with receiving a bug report through the activity 'receive bug report,' because of the init constraint." In particular, all 16 subjects (100.0 %) started describing the process models at an activity with the init constraint.

Afterward, 12 out of 16 subjects (75.0 %) working on process models with the normal layout continued describing the model as stated in Sect. 3.3.1, i.e., after mentioning the entry point, they tried to figure out in which order the activities are to be executed. In particular, this strategy was observed 8 out of 9 times (88.9 %) for P_{5_normal} and 4 out of 7 (57.1 %) times for P_{6_normal}. In addition, we could observe that for P_{6_normal}, some subjects seem to prefer starting their analysis with components of low complexity before continuing with more complex components (3 out of 7, 42.9 %). Moreover, for P_{5_normal}, one subject had a strategy differentiating between unary constraints (e.g., existence constraints) and binary constraints (e.g., relation and negation constraints). In particular, after mentioning the init constraint, the subject analyzed all existence constraints, followed by all remaining binary constraints. When analyzing the binary constraints, the subject analyzed the model bottom-up following precedence constraints constituting preconditions for activity execution.

In addition, we could observe that subjects working on process models with normal layout described the model at different granularity levels. The majority of subjects, i.e., 9 out of 16 subjects (56.3 %), described the model at a fine-grained level and mentioned most or all of the activities and also considered dependencies between constraints. For P_{5_normal}, this was the predominant strategy (7 out of 9 subjects, 77.8 %). Second, 5 out of 16 subjects (31.3 %) read (parts of) the process models by *building blocks*, rather describing roughly the goal of the components than reading each activity label: ("This part describes relations with partners or something like that…"). Interestingly, this strategy only occurred for P_{6_normal} (5 out of 7 subjects, 71.4 %). Finally, 2 subjects described P_{5_normal} only very roughly by either only mentioning activity labels (1 subject) or by focusing on the overall intention of the process model (1 subject), i.e., "…the process seems to be a developing process in the context of electronic devices."

Mirrored declarative process models As normal laid out models have a clear trend how subjects read them, we investigated whether and how subjects are influenced by a changed model layout. For example, Fig. 7 shows $P_{6\_mirrored}$. The model consists of six components. The first one contains activities "attend staff briefing" and "fetch job cards." The second component consists of the single activity "answer questions of apprentices." Three components comprise activ-

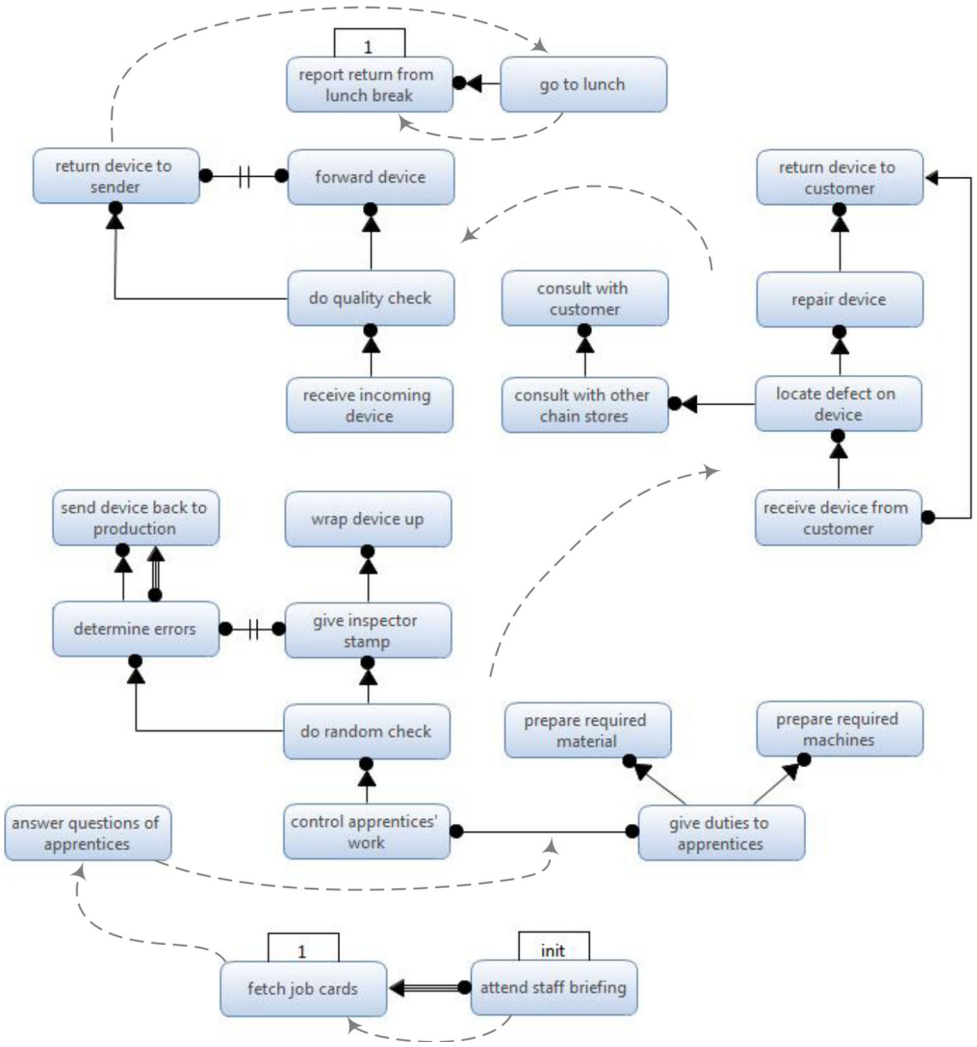


Fig. 7 $P_{6\_mirrored}$ and a reading variant

ities related to the typical workload of a worker of an electronic company. The last component contains the activities "go to lunch" and "report return from lunch break."

The graph analysis of $P_{5\_mirrored}$ and $P_{6\_mirrored}$ indicates different strategies how subjects read these models (cf. dotted arrows in Fig. 7 for an example of how a subject tried to make sense of the process models). Most frequently (like for the normal layout), subjects started their analysis with the initial activity of the model (9 out of 16 subjects, 56.3 %). In particular, this was a popular strategy for $P_{5\_mirrored}$ (5 out of 7 subjects, 71.4 %). For $P_{6\_mirrored}$, only 4 out of 9 subjects (44.4 %) started at the activity with the init constraint: "Ok, we have an init activity here, so we start with 'attend staff briefing'." Compared to the model with the normal layout, this strategy was less frequently applied. We could observe that it sometimes took subjects a while to identify the entry point (1 subject for $P_{5\_mirrored}$): ("Here there are around 10 or 11 activities. Mmm…There are some isolated activities, for example 'ask supervisor for help', this is completely isolated. […] In fact there is one activity that has an init relation, hence, it has to be the first activity to be executed in the complete process"). Having found the entry point, subjects typically described the rest of the process as described in Sect. 3.3.1, but in a mirrored manner (i.e., mentioning the activities in the order they are presumably executed), as depicted in Fig. 7. Overall, 6 out of 7 (85.7 %) subjects followed this strategy for $P_{5\_mirrored}$. Moreover, 2 out of 9 subjects (22.2 %) used this strategy for $P_{6\_mirrored}$. One of the 9 subjects (11.1 %) starting the analysis at the init constraint deviated from this pattern. After analyzing the init constraint, the subject first looked at the small components before analyzing the more complex ones. Within each component, however, the subject tried to consider the execution order of activities.

Another re-occurring strategy was to start the analysis in the upper left corner, i.e., with activity "insert bug report into

archive" in $P_{5\_mirrored}$ and with "return from lunch break" in $P_{6\_mirrored}$. For example, one subject uttered "I see at the top of the model an activity that needs to be executed once…it's about lunch." Afterward, the subjects continued the analysis in a top-down manner (4 out of 16, 25.0 %; 1 out of 7 for $P_{5\_mirrored}$, 14.3 %, and 3 out of 9 for $P_{6\_mirrored}$, 33.3 %).

For $P_{6\_mirrored}$ 3 out of 9 subject (33.3 %), we could not identify a systematic reading strategy; subjects seemed to name activity labels in a random manner. Respective subjects mainly tried to make sense of $P_{6\_mirrored}$ and to come to a conclusion regarding the overall intent of the process by focusing by reading activity labels (mostly ignoring constraints) and by relying on the domain knowledge.

Again, we could observe that subjects described the model at different granularity levels. Similar to the normal layout, the majority of subjects, i.e., 10 out of 16 subjects (62.5 %), described the mirrored process models at a fine-grained level and mentioned most or all of the activities and constraints. While for $P_{5\_mirrored}$, all subjects (7 out of 7, 100.0 %) followed this strategy, only 2 out of 9 subjects (22.2 %) described $P_{6\_mirrored}$ at a fine-grained level. For $P_{6\_mirrored}$, 2 subjects read the model by building blocks roughly describing the goal of the component, 5 subjects mostly focused in their analysis on activity labels to determine the overall goal of the process.

Discussion In the exploratory study, we found that subjects tend to start reading the process model at the init constraint and to sequentially go through the process model in the order activities might get executed. Since the init constraints in the models used in the exploratory study were in both cases in the upper left corner of the model, we could not determine whether the results of the exploratory study were due the init constraint being an important factor or due to the fact that people tend to start reading at the top [19]. To eliminate the impact of the process model's layout, we used two model variants in the follow-up study, a normal layout (similar to the exploratory study) and a mirrored version. We expected that if the order of activities determines the reading direction of subjects, that subjects start their analysis of the mirrored models at the bottom of the model (with the init constraint) and not in the upper left corner. If, however, the prevalent reading direction in our culture (from left to right) is the determining factor, the analysis of mirrored models has to start in the upper left corner and not with the init constraint. Our results suggest that init constraints (9 out of 16 for $P_{5\_mirrored}$ and $P_{6\_mirrored}$, 56.3 %) seem to be more important than the reading direction (4 out of 16 for $P_{5\_mirrored}$ and $P_{6\_mirrored}$, 25.0 %).

In addition, we could identify different strategies people apply for analyzing a declarative process model once an entry point has been found. For the models with normal layout, the most common strategy in the exploratory study was to analyze models top-down (considering at the same time the execution order of activities). This could be replicated for the models with the normal layout in the follow-up study. For the mirrored process models, however, 8 out of 16 (50.0 %) subjects analyzed process models bottom-up considering the execution order of activities and 4 out of 16 (25.0 %) read the model top-down in the typical reading direction. This again suggests that starting with the init constraint was more important to the subject than the prevalent reading direction of our culture.

In terms of description granularity, we could observe considerable differences between P_5 and P_6, irrespective of the models' layout. While for P_5 the majority of subjects described the model at a fine-grained level, i.e., at the level of single activities (14 out of 16, 87.5 %), descriptions at the level of building blocks were frequently used for P_6 (7 out of 16, 43.8 %). Another frequently used strategy for P_6 was the focus on activity labels (5 out of 16 subjects, 31.3 %). A possible explanation for the differences between P_5 and P_6 might be the differences in the complexity of components between P_5 and P_6. P_5 only contains 3 components including a rather complex one with 9 activities, 15 constraints, and 7 different constraint types. P_6, in turn, contains 6 components (with at most 9 activities and 10 constraints, 4 different constraint types). The fact that P_6 contains more, but smaller components suggests that it was easier for subjects to aggregate model information to building blocks, therefore applying this strategy more frequently. However, from the data, we cannot conclude this with certainty making a follow-up investigation necessary. Another open question to be answered in a follow-up question is why labels are more frequently used for P_6.

4.3.2 $RQ_{2.1}$: *What are the challenges system analysts are facing when performing basic model interpretation tasks, such as determining minimal traces, valid traces, and invalid traces?*

As detailed previously, the subjects were asked to identify 2 invalid traces for each model, the minimal trace, and 2 valid traces. Since 16 subjects participated in the study and each subject worked on two process models, 32 answers were collected regarding the minimal trace (16 for each model). Further, 64 for valid traces (32 for each model) and 64 invalid traces (32 for each model) were collected. Figure 8 illustrates the distribution of correct and incorrect answers for invalid traces, minimal traces, and valid traces, respectively. Further, Fig. 8 shows a categorization of the different types of errors and their frequency.

Invalid traces Questions requiring the identification of invalid traces were answered correctly to a large extent, i.e., 62 out of 64 questions (96.9 %) were answered correctly. This seems

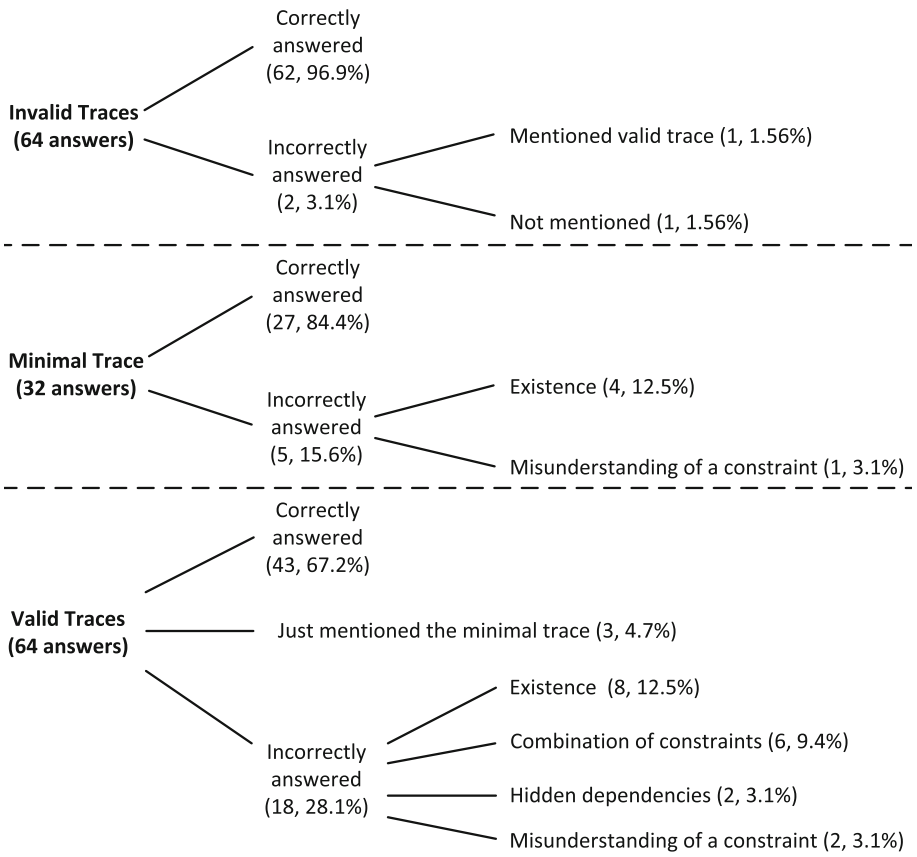


Fig. 8 Distribution of errors when naming minimal, valid, and invalid traces

reasonable since invalid traces can be constructed by violating a single constraint of the model. Both errors regarding invalid traces were committed by the same subject. First, the subject named the minimal trace and after that did not give another invalid trace.

Minimal trace To identify the minimal trace, the subjects had to find the initial activity, which was determined for both models by init constraints, and check the effects of executing this initial activity. For example, if the initial activity is part of a response constraint, which was the case in both models, the response constraint forces the execution of another activity. Additionally, activities with existence constraints must be considered, which, in turn, might enforce the execution of related activities. 5 out of 32 answers (15.6 %) were incorrect. 4 incorrect answers (12.5 %) are related to the violation of existence constraints. For example, for P_5, one subject uttered "The minimal trace... 'Receive bug report', 'search for bug in archive', minimal... Ok. That's all," forgetting about the existence constraint on activity "try reproduction." Similarly, for P_6, one subject uttered "we have to start with 'attend staff briefing' and just after that we have to 'fetch job cards'. This can be a minimal trace... the minimal trace can be that," forgetting about the existence constraint on activity "report return from lunch break." A possible explanation could be that either some subjects misunderstood existence constraints. Alternatively, they might have overlooked existence constraints. One incorrect answer was given by a subject who confused a precedence with a response constraint, which forced the execution of another activity. Specifically, the subject included the activity "ask user" after the activity "try reproduction" as part of the minimal trace of P_5.

Valid traces The subjects were asked to identify two valid traces apart from the minimal trace. Consequently, the subjects are required to add additional activities to the minimal trace. When adding these activities, the constraint have to be examined to find potential consequences of adding activities, e.g., response constraints. We observed the highest number of incorrectly answered questions for this task. Specifically, 18 out of 64 (28.1 %) answers were incorrect. Additionally, the minimal trace was given 3 times, even though we explicitly asked for different traces. Most errors, i.e., 8 out of 18 (12.5 % overall), are caused by subjects ignoring existence constraints. We identified a fairly large overlap with errors on minimal traces. 6 out of the 8 errors were committed by the same subjects as for minimal traces. Further, for those 6 subjects, the same activity was missed in the valid traces. This seems reasonable since most subjects took the minimal trace and added some activities to form examples of valid

traces. Therefore, subjects, who forgot certain activities in the minimal trace, tended to forget the same activities when naming examples of valid traces. For example, for P_5, one subject uttered "The first trace is going to be adding ask supervisor to the minimal trace, I mean: 'receive bug report', 'search for bug in archive', and 'ask supervisor'," forgetting about the existence constraint on activity "try reproduction." In a similar way, for P_6, one subject uttered "ok... again we start with 'attend staff briefing', 'fetch job cards', and ... and ... we can execute for example 'give duties to apprentices' and 'control apprentices work'," forgetting about the existence constraint on activity "report return from lunch break." Several errors, i.e., 6 out of 18 (9.4 % overall), were committed by subjects having problems with the combination of constraints, all related to P_5. For example, one subject uttered "Other trace would be 'receive bug report', 'search for bug in archive', 'try reproduction', and 'write a test'," forgetting to include activity "run test" as required by the constraints between "write test" and "run test." Furthermore, 2 incorrect answers were given in P_6 due to subjects having problems with hidden dependencies. For example, one subject uttered "And another one (trace) can be 'attend staff briefing', 'fetch job cards', 'receive device from customer', 'return device to customer', 'go to lunch' and 'report return from lunch break'," leaving out the activities "locate defect on device" and "repair device." Lastly, 2 answers were incorrect in P_6 due to misunderstanding constraints. Specifically, one subject uttered "One valid trace could be 'fetch job cards', 'attend staff briefing', 'answer questions of apprentices', 'go to lunch' and 'report return from lunch break'," which indicates that the subject misunderstood the chained response constraint between activities "attend staff briefing" and "fetch job cards," i.e., the ordering of activities was not correct.

Discussion In general, we observed only very limited difficulties when subjects were asked to name invalid traces. This is not surprising as invalid traces can be constructed by selecting a single constraint that can be violated. For instances, P_6 contains the activity "fetch job cards" (cf. Fig. 7), which has to be executed exactly once. An invalid trace can therefore be achieved by including this activity twice in the trace. This finding seems reasonable when considering the task with a cognitive background. Forming an invalid trace can be achieved by utilizing a small portion of the process model, occupying only limited space within working memory. Therefore, the task can be accomplished without overstraining the working memory's capacity [15,37], resulting in a small number of errors [60]. We observed considerably more difficulties when subjects were asked to name the minimal trace for the process model. This seems reasonable, as a higher burden is put on the subject's working memory, i.e., more elements need to be maintained in working memory to identify the minimal trace. This problem is further amplified when asking the subjects for valid traces apart from the minimal trace, i.e., even more elements need to be maintained in working memory. Regarding the naming of minimal and valid traces, the most common cause of error was related to subjects ignoring the existence constraints. This number is put into perspective for valid traces by the overlap of errors between naming minimal traces and valid traces, i.e., subjects who missed an activity in the minimal trace committed the same error for valid traces. Keeping this in mind, it seems that the combination of constraints becomes a more dominant issue for forming valid traces. This seems reasonable, as more elements need to be integrated in working memory while remembering the consequences of including an additional activity in the trace, i.e., putting an additional burden on the subject's working memory.

4.3.3 $RQ_{2.2}$: What are the challenges that arise when system analysts have to deal with pairs of constraints between activities?

To target this research question, we investigated how subjects deal with pairs constraints between two activities while making sense of a process model. In particular, two pairs of constraints were introduced in P_5 and one pair was introduced in P_6—in the following, we will describe how subjects approached these situations.

First, we turn toward model P_5, in which a pair of constraints between the activities "write tests" and "run test" can be found (cf. Fig. 6). While the precedence constraint ensures that it is not possible to run a test before it was written, the chained response constraint requires a test to be run directly after it was written. As discussed in RQ_1, 2 out of 16 subjects (12.5 %) did not mention constraints on detail, but rather focused on activity labels. The remaining 14 subjects (87.5 %) looked at this pair of constraints in detail, and thereof, 13 subjects (81.3 %) had no problems understanding this pair of constraints, e.g., "regarding the activity 'run test', before we are able to execute 'run test', we need to write a test and just after finishing 'write test' we need to execute 'run test'." Still, one subject (6.3 %) mentioned that the precedence and chained response constraints should be merged into a chained succession: "These 2 constraints are not combined in a nice way. Because in one case there is a precedence and in the other one there is a response. If we want to take them in both ways, this would be a succession. In fact, this would be a chained succession." However, merging both constraints into a single-chained succession would remove the possibility of executing the test whenever it is needed, e.g., after executing activity "write code." Hence, we conclude that subjects did not reveal considerable problems with this pair of constraints.

Second, we look into another pair of constraints in P_5. In particular, as shown in Fig. 6, between "run test" and "write code," a chained precedence constraint and a chained response constraint exist. Hence, before code can be written, the test must be run immediately (chained precedence), and after writing code, the test must be run immediately (chained response). Again, 14 out of 16 subjects (87.5 %) looked at this pair of constraints in detail. 5 subjects (31.3 %) interpreted the combination incorrectly. Particularly, 2 subjects claimed that directly after "run test," "write code" must be executed, i.e., confusing the chained precedence with a chained response constraint: "And when 'run test' finishes, 'write code' has to be executed." 2 more subjects said that after "write code," "merge fix" has to be executed: "After writing code, this fix must be merged and tested." One subject interpreted the constraints between "write code" and "run test" correctly, but got confused to point where the subject argued that "they can't be executed.". Hence, compared to the first situation of paired constraints, more subjects experienced problems with the pair of constraints.

Third, in P_6, activities "determine errors" and "send device back to production" are connected with a precedence constraint and a chained response (cf. Fig. 7). Therefore, it is not possible to send a device back to production before determining its errors and a device must be sent back to production directly after its errors were determined. While describing P_6, 6 out of 16 subjects (37.5 %) mentioned these two activities (the other subjects described the model on a more abstract level, cf. 4.3.1). Among these 6 subjects, only two explicitly mentioned both constraints, i.e., the precedence and chained response constraint. One of them understood the situation after taking some time. The second one stated that these two constraints should be combined to a chained succession: "in fact I think it is the same case, they could be combined in a chained succession." Again, merging both constraints into a single-chained succession would remove the possibility of sending a device back to production whenever it is needed. In short, only 10 subjects looked at constraints at all, 4 subjects only mentioned one constraint, 1 subject understood the combination correctly, and 1 subject showed problems. However, this subject was the same who misinterpreted the first pair of constraints in P_5.

Discussion The findings obtained in $RQ_{2.2}$ indicate that pairs of constraints can cause considerable problems when making sense of declarative process models. In particular, for a pair of constraints in P_5, error rates of 31.3 % could be observed. However, for the remaining, two pairs—except for 2 misinterpretations of the same subject—were interpreted correctly. Hence, we argue that pairs of constraints can pose considerable problems, but the exact causes call for further investigation.

4.3.4 $RQ_{2.3}$: *What are the challenges that arise when system analysts have to deal with hidden dependencies?*

To investigate this research question, we asked subjects two questions regarding hidden dependencies for model P_5 and model P_6 (16 subjects, 2 questions per model, 2 models, resulting in 64 answers). Figure 9 shows the distribution of answers: Overall, subjects answered 50 out of 64 questions correctly (78.1 %).

Regarding incorrect answers (14 out of 64), 12 (18.8 %) answers were wrong because subjects did not look at the connections between the activities of the model close enough, i.e., overlooked a hidden dependency. For example, when asked for process model P_5 whether ⟨"search for bug in archive," "ask user"⟩ is a valid sub-trace, several subjects overlooked the hidden dependency between activities "receive bug report" and "search for bug in archive" (cf. Fig. 6). One subject uttered that "if we add some activities, this sub-trace can be a valid trace[…] Adding 'receive bug report' and 'try reproduction'… this will be a valid sub-trace, so I think it's true." However, adding the two mentioned activities before the given sub-trace is not possible due to the chained response constraint between activities "receive bug report" and "search for bug in archive," requiring that activity "search for bug in archive" is executed directly after activity "receive bug report." In one case, the subject answered the question incorrectly, because of confusing a precedence constraint with a chained response constraint. Thus, the problem was not really caused through hidden dependencies, but a lacking knowledge of constraint semantics. For one subject, the error source could not be determined, since no reasoning was provided.

Results indicate that hidden dependencies constitute a considerable challenge for system analysts and are frequently overlooked. However, it is unclear whether this is because subjects not being aware of hidden dependencies or whether this is because such dependencies can be easily overlooked. Therefore, as summarized in Fig. 10, after questions regarding the pitfalls of Declare models, we asked subjects whether they could recognize any hidden dependency while describing the model and answering questions about it.

Overall, subjects reported from 52 alleged hidden dependencies, of which 44 were indeed proper hidden dependencies. Interestingly, only 4 subjects named only correct hidden dependencies. In addition, 6 subjects named at least one correct hidden dependency. One subject, in turn, only mentioned incorrect hidden dependencies. For example, one subject pointed out correctly "another example for a hidden dependency would be precedence constraints that result from the init constraint and stretch across the whole model," i.e., an init constraint implies a precedence relation to all other activities. In several cases, our transcripts revealed that not

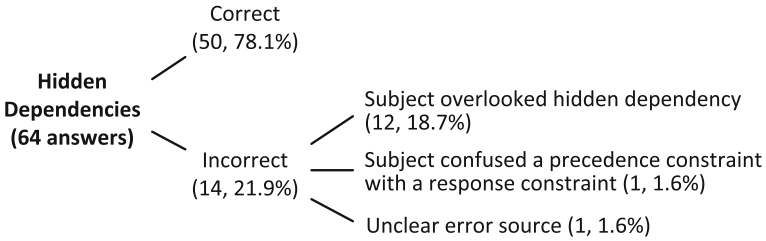


Fig. 9 Distribution of answers for questions regarding hidden dependencies

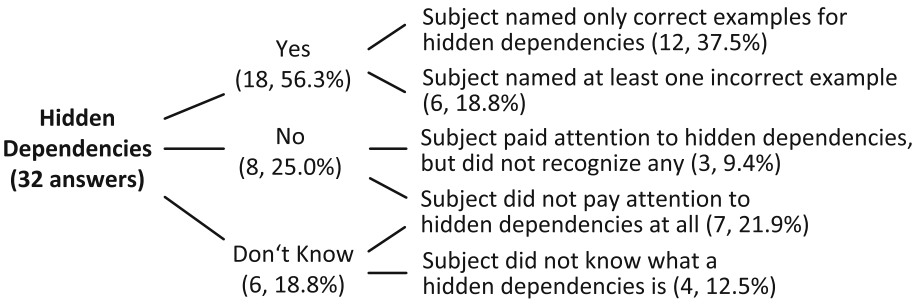


Fig. 10 Distribution of answers regarding hidden dependencies

all subjects really know what hidden dependencies are. In particular, 8 incorrect hidden dependencies were mentioned, seemingly because the subjects had problems to understand difficult situations ("Every time I execute 'try reproduction' I have to execute this sort of loop… this is a hidden dependency"). Moreover, 5 subjects did not mention any hidden dependencies, either because they did not know what hidden dependencies are (2 subjects) or because they did not look out for hidden dependencies or could not find any (3 subjects). For example, one subject stated that "until now I didn't recognize any hidden dependencies…I didn't pay attention to this aspect."

Discussion The follow-up study showed that hidden dependencies (like the combination of constraints) cause considerable difficulties to subjects. When asked questions regarding two models including hidden dependencies, subjects only answered 78.1 % of the questions correctly. Asked to identify hidden dependencies for P_5 and P_6, the percentage of correct answers was even lower: only 4 out of 16 subjects (25.0 %) were able to only name correct hidden dependencies. Knowing that the identification of hidden dependencies requires the extraction of implicit information, i.e., information that needs to be computed in the human mind [56], it seems plausible that the identification of hidden dependencies is indeed a difficult task.

4.3.5 $RQ_{2.4}$: *Do challenges arise when system analysts have to deal with existence constraints? If yes, how do system analysts deal with these challenges?*

As detailed previously, 2 questions per model were related to the number executions of certain activities (i.e., each subject answered 4 questions focusing on existence constraints). The

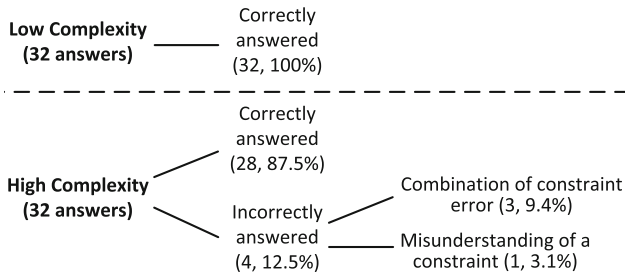


Fig. 11 Distribution of errors for questions on existence constraints

questions were designed to focus on two different aspects: (1) questions of low complexity, which require simple reasoning (i.e., the subjects only need to analyze one activity and the corresponding existence constraint for answering the question) and (2) questions of higher complexity, which require a more complex reasoning process (i.e., the subjects need to analyze several activities and constraints together with potential hidden dependencies for answering the question). Since 16 subjects participated in the study and 2 questions were included in each aspect, i.e., low/high complexity, 32 answers were recorded for each aspect. Figure 11 illustrates the percentage of their correct and incorrect answers together with the distribution of errors.

For questions of low complexity, the subjects did not experience any difficulties understanding existence constraints since all questions were answered correctly. This makes it reasonable to assume that the errors subjects made when naming minimal and valid traces (cf. Sect. 4.3.2) are rather due to overlooked existence constraints than misunderstood existence constraints. In contrast, when analyzing the results of questions of medium complexity, several errors occurred due to misunderstood constraints. Specifically, for P_5, 2 subjects thought that chained precedence(A,B) together with

Table 3 Modeling experience and education versus accuracy and mental effort

Factor	Declarative		Imperative	
	Acc.	M. eff.	Acc.	M. eff.
Years of modeling experience	0.414	−0.248	0.250	−0.049
Formal training last year	0.150	−0.068	−0.030	−0.130
Self-education last year	0.270	0.264	0.069	−0.023
Amount of analyzed models	**0.756****	−0.249	0.254	−0.158
Amount of created models	**0.657****	−0.351	0.208	0.082
Avg. activities per model	**0.555***	**−0.528***	−0.035	−0.384
Familiarity with paradigm	**0.537***	**−0.518***	0.034	−0.170
Confidence in understanding	**0.604***	**−0.566***	−0.013	−0.196
Competence in modeling	0.393	−0.484	−0.225	−0.260

* Significant at the 0.05 level
** Significant at the 0.01 level

max(B,1) implies max(A,1), which is not the case. For example, one subject uttered "It is possible that 'test fix' is executed ten times in a process instance… chained precedence […] 'test fix' must be executed immediately before 'commit changes' and 'commit changes' can be executed only exactly once,… in this case I think it is false that it can be executed ten times, because of that relationship between 'test fix' and 'commit changes'." In a similar way, for P_6, 1 subject thought that precedence(A,B) together with exactly(B,1) implies exactly(A,1), which is not the case. For example, one subject uttered "because the activity 'report return from lunch break' can be executed only once, 'go to lunch' can also be activated only once in a process instance." The other subject committing an error for P_6 confused the meaning of a constraint.

Discussion When asking the subjects for existence constraints, we observed only a limited amount of difficulties; especially, when asking for existence constraints that are not related to other activities via combinations of constraints, we did not observe a single error. When requiring more complex reasoning for answering the question, we observed three errors that can be attributed to difficulties with the combination of constraints, i.e., the subjects derived an incorrect meaning from the constraint combination. Therefore, we conclude that existence constraints cause only a limited amount of difficulties. Further, when problems arise, they are rather caused by combinations of constraints than the existence constraint itself.

4.3.6 $RQ_{3.1}$: *Are modeling experience and education connected to a lower error rate in interpreting declarative process models?*

So far, we investigated problems associated with the sense-making of declarative process models. Next, we turn toward ways for supporting the sense-making of declarative process models. Generally, education has a positive influence on the understanding of imperative process models [35,54]. In this sense, it appears likely that the same holds for declarative process models and that the influence of education related to declarative process modeling is beneficial for the understanding of declarative process models. However, it is not clear whether also education related to imperative process modeling helps in the sense-making of declarative process models. To this end, we screened the education and experience regarding declarative and imperative process modeling and correlated the results with accuracy and mental effort, i.e., the understanding of the process models.

The results of this analysis can be found in Table 3. In particular, the first column lists factors regarding education or experience, whereas columns two to five show Spearman's rho for accuracy and mental effort, thereby columns two and three focus on education and experience regarding declarative process modeling, while columns four and five focus on education and experience regarding imperative modeling. Apparently, a distinction between education and experience regarding declarative and imperative modeling can be made. More specifically, the amount of analyzed and created declarative models correlates statically significant with accuracy, i.e., the amount of correct answers. Furthermore, average activities per declarative model, self-rated familiarity with declarative modeling in general, and self-rated confidence in understanding declarative models correlate statistically significant with accuracy and mental effort. Contrariwise, *none* of the factors of educational background regarding imperative modeling shows a statistically significant influence on accuracy or mental effort.

Having established that declarative background has an influence on the amount of correct answers, the question arises whether similar errors are conducted by persons with different levels of declarative knowledge and experience. To investigate this question, we used the following procedure. First, we computed a declarative knowledge score for

Table 4 Correct answers per group and question category

Category	Lower declarative knowledge	Higher declarative knowledge
Traces	59	61
Paired constraints	19	27
Hidden dependencies	19	21
Existence constraints	31	30

each subject by aggregating variables that showed significant correlations with error rates, i.e., amount of analyzed models, amount of created models, average activities per model, familiarity with paradigm, and confidence in understanding (cf. Table 3). To compensate for different levels of measurement, e.g., amount of created models versus familiarity with paradigm, we created rankings for each of the variables according to the ranking method used in Spearman's rank correlation coefficient. Then, we summed up these rankings for each background variable, denoted as declarative knowledge score. Based upon this score, we divided our sample into two groups, i.e., a group with 8 subjects showing a declarative background knowledge below average and 8 subjects showing a declarative background knowledge above average.

The results of this procedure are summarized in Table 4: the columns show the groups with lower and higher declarative background knowledge, whereas the rows list the investigated categories. The numbers suggest that similar numbers of correct answers, except for category paired constraints, were found. To test whether differences between categories exist, we conducted Mann–Whitney U test between the group with lower declarative knowledge against the group with higher declarative knowledge for all categories: traces ($U = 17.5$, $p = 0.111$), paired constraints ($U = 17.0$, $p = 0.067$), hidden dependencies ($U = 23.0$, $p = 0.289$), and existence constraints ($U = 32.0$, $p = 1.000$). Even though none of these comparisons could be found to be statistically significant at α of 0.05, the computed p values differ considerably. In particular, virtually no difference could be found for existence constraints ($p = 1.000$), whereas differences approached the significance niveau for paired constraints ($p = 0.067$) and traces ($p = 0.111$). In other words, it can be assumed that existence constraints cause similar difficulties for inexperienced and experienced subjects. However, paired constraints and traces seem to be particularly more difficult for less-experienced subjects and hidden dependencies range between these categories. We would like to emphasize at this point that these results do *not* suggest that there are no differences between less and more experienced subjects, but rather the low sample size might be responsible for nonsignificant differences. Nevertheless, even if the frequencies of some of these difficulties would be proven different between experienced and inexperienced subjects in larger samples, our finding that both populations encounter these difficulties, to a certain extent, remains valid.

Discussion Considering these results, it seems essential that system analysts that are introduced to declarative modeling receive adequate training and that the influence of knowledge from imperative modeling is rather limited. Likewise, regarding the recent interest in modeling languages that combine declarative and imperative modeling constructs [55], appropriate training seems essential. This finding is particularly relevant, as it was found that within imperative modeling languages—particularly EPC and BPMN—it does not matter which modeling language is taught [51]. Against this background, it seems that *within* the same modeling paradigm, knowledge can be rather easily transferred. However, in our data, no statistically significant correlations between imperative modeling knowledge and the understanding of declarative models could be found.

4.3.7 $RQ_{3.2}$: Is there a relationship between the number of correctly answered questions and the mental effort spent?

Regarding $RQ_{3.1}$, we investigated in how far human-related factors, such as experience and education, are connected to the understanding of declarative models. Further, $RQ_{3.2}$ is concerned with supporting the sense-making of declarative models, in particular with the question how improvements can be measured. In this vein and detailed in the following, works investigating potential problems regarding the understanding of declarative models referred to problems closely connected to the capabilities of the human mind. For instance, the combination of constraints [47], hidden dependencies [39,67], and hard mental operations [67] was identified as potential problems. Therefore, it seems desirable to have measures at hand that allows researchers to assess in how far proposed concepts support the human mind in interpreting declarative models. To this end, as described in Sect. 2.5, the measurement of mental effort seems to be promising, as it presumably allows assessing subtle changes with respect to understandability [65]. However, currently, it is not clear yet whether mental effort is indeed a useful measure for the understandability of a declarative model. To compensate this shortcoming, in the following, we investigate the connection between mental effort and accuracy, i.e., the percentage of correct answers—an established measure for understandability [2].

To this end, we computed the average mental effort and accuracy for each question. As described in Sect. 4.1, we prepared 2 models with 8 questions each, leading to a total of 16 questions. To visualize the results, we employed a scatter plot, as shown in Fig. 12. The x axis represents the men-

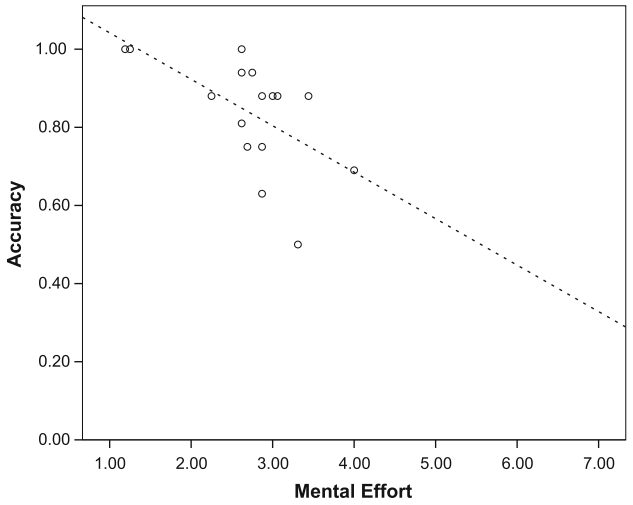


Fig. 12 Mental effort versus accuracy

tal effort, ranging from *Extremely low mental effort* (1) to *Extremely high mental effort* (7). In turn, the y axis shows the associated accuracy, ranging from 0 (all answers incorrect) to 1 (all questions correct). Furthermore, three observations can be made. First, the mental effort of most questions is below *Neither high nor low mental effort* (4), i.e., subjects perceived the questions to be rather easy. In fact, the questions were ranked to have on average *Low mental effort* ($M = 2.71, \text{SD} = 0.71$). Second, questions were mostly answered correctly, i.e., 84.0 % of all answers were correct ($M = 0.84, \text{SD} = 0.15$). Third, it appears that, in general, higher mental effort is connected to lower accuracy. To corroborate these observations, we computed Spearman's rho, confirming that mental effort and accuracy are negatively correlated [$r_S(14) = -0.628$, $p = 0.009$].

As opposed to this background, the adoption of mental effort for measuring the understandability of a declarative process models seems to be promising. To provide further support for the usefulness of mental effort, we need to refer to an earlier experiment, in which we investigated the impact of test cases on the maintenance of declarative process models [63], thereby we assessed mental effort, accuracy, and confidence—typical measures regarding understandability [2]. Even though a positive influence on mental effort and confidence could be found, accuracy did not change. In the discussion, we argued that nonsignificant differences could be traced back to problems with the experimental design. In fact, the replication of the study confirmed the positive influence on mental effort and confidence, but also on accuracy [65]. Hence, we argue that the adoption of mental effort provides a valuable additional perspective, allowing us to assess in more detail which research directions seem most promising.

4.3.8 $RQ_{3.3}$: Which modeling constructs are perceived particularly challenging and where do system analysts see a potential for improving the understandability of declarative process models?

The goal of $RQ_{3.3}$ was to complement findings obtained so far with opinions and suggestions from system analysts. In particular, after all questions regarding the understandability of the process model were answered, we additionally asked the following questions for each model:

- Why do you think the model was (not) difficult to understand?
- Do you have suggestions for making the model easier to understand?

To analyze answers, we once more applied grounded theory to identify and classify issues, which—according to the subjects—influence the sense-making of declarative business process models. The results for the question "Why you think the model was (not) difficult to understand?" are summarized in Table 5. All in all, we could find 9 factors that subjects considered to be beneficial/harmful for the sense-making of declarative process models. These factors, in turn, were grouped into three categories, i.e., factors relating to constraints, factors relating to the imperative background of the subjects, and other factors. Considering category *constraints*, 16 subjects mentioned that the combination of constraints posed a considerable challenge for the sense-making. For instance, one subject mentioned that "for me it is difficult to understand the meaning… since we have several relations, several constraints which relate the order of the activities." In addition, 4 subjects explicitly mentioned that they experienced problems with hidden dependencies. Positively, 4 subjects explained that components supported the sense-making. Knowing that almost all subjects had problems with the combination of constraints, this appears plausible as components help to reduce the amount of activities connected by con-

Table 5 Why do you think the model was (not) difficult to understand?

Category	Factor	Subj.	Infl.
Constraints	Combination of constraints	16	–
	Hidden dependencies	4	–
	Components	4	+
	Semantic of constraint	4	–
Imperative background	Components	4	–
	Cycle	4	–
	Init	2	+
	Flow	1	+
Other	Layout	4	–

straints. Interestingly, only 4 subjects mentioned that they had problems with the semantic of single constraints: precedence, response, succession, and all chained constraints were mentioned. As *all* subjects indicated problems with the *combination* of constraints, this clearly indicates problems are rather caused by the combination of several constraints than by the semantic of *single* constraints.

Regarding category *imperative background*, we could find two negative and two positive influences. Interestingly, 4 subjects mentioned components as negative influence—even though another 4 subjects deemed components as a positive influence, as described before. Also, 4 subjects indicated constraints that appeared visually, but not necessarily semantically, as cycles, as problem. To understand these peculiar and apparently contradicting findings, we would like to refer to the theory of *Mindshift Learning* [3]. This theory postulates that, when learning new modeling languages, concepts that are similar, but still show subtle differences, are most difficult to learn. In the context of imperative and declarative process modeling languages, one particular problem is related to the graph-based notation. Even though both paradigms typically make use of graph-based notations, the semantics are usually different to a large extent. Regarding components, in an imperative process model, related activities are usually connected be sequence flows. In declarative process models, however, related activities do not necessarily need to be connected by constraints. Likewise, cycles in imperative process models are created through sequence flows. In declarative process models, constraints do not necessarily convey sequential information only; thus, activities that are visually connected as a cycle do not necessarily describe a cycle semantic-wise. In light of the theory of Mindshift Learning, these concepts are similar, but yet not the same and thus particularly difficult to understand. Hence, it appears plausible that components and cycles were mentioned as factors negatively influencing sense-making. Contrariwise, we also found two factors that were perceived positively. First, 2 subjects mentioned that the init constraint—which relates to a start event in an imperative process model—is useful in the interpretation. Likewise, 1 subject mentioned that it appreciated when the order in which activities are to be executed matches the layout. Finally, regarding category *other*, we would like to mention that 4 subjects perceived the layout as a negative influence. However, this can be rather traced back to the setup of the study, i.e., the usage of mirrored layout, than the specificities of Declare models.

Finally, question "Do you have any suggestions to make the model easier to read or understand?" gives insights into what subjects proposed to make declarative process models easier to understand. In particular, as summarized in Table 6, four suggestions were identified. First and foremost, 16 subjects urged that the combination of constraints needs to be simplified. Further, 6 subjects proposed the development of mechanisms that make hidden dependencies explicit. In addition, 6 subjects indicated that a layout that aligned activities in their execution order from left to right would be beneficial. Finally, 5 subjects proposed to make use of modularization for improving the understandability of declarative process models.

Table 6 Do you have suggestions for making the model easier to understand?

Suggestion	Subjects
Simplify combination of constraints	16
Make hidden dependencies explicit	6
Change layout (from left to right)	6
Use modularization	4

Unsurprisingly, suggestions for the improvement of declarative process models are closely connected to respective problems (cf. Table 5). In general, it can be observed that the basic building blocks of declarative process models—activities and constraints—are rather unproblematic. However, the combination of constraints and resulting hidden dependencies, in turn, pose considerable challenges. In this sense, for instance, approaches providing computer-based support for the interpretation of constraints seem promising [67]. Also, regarding the design of declarative process modeling languages, it seems advisable to refrain from using modeling constructs that look visually similar to imperative modeling constructs, but exhibit different semantics—such modeling constructs are particularly difficult to learn, as argued in the Mindshift Learning theory [3].

4.4 Limitations

The findings of this study have to be seen in light of several limitations. First and foremost, due to the similar nature of experimental design, limitations of the exploratory study also apply to the follow-up study. This includes the relatively low sample size (even though not unusual [14,43]). Further, although the adopted declarative process models vary in the number of activities, constraints, domain and layout, it is not entirely clear whether the results can also be applied for declarative process models in general. Likewise, all subjects indicated academic background, further limiting the generalization of results. Lastly, note that this study focuses exclusively on Declare models and further studies are needed to establish whether the same conclusions would apply to other declarative process modeling languages.

5 Related work

In this work, we have empirically investigated the sense-making of Declare models. Similarly, the connection between

a declarative process model's modularization and its understanding was empirically validated in [68]. However, as opposed to this work, the focus was put on modularization further than on the general understanding of a declarative process model. More generally, the understandability of conceptual models with hierarchy was investigated for a variety of modeling languages, such as PROTOS [53], Hierarchical ER Diagrams [40], and UML Diagrams [10] (see [64] for an overview). Similarly, the understandability of conceptual modeling languages was examined from different angles. For instance, the understandability of Extended ER Diagrams and the Nijssen Information Analysis Methodology was compared in [31], whereas the understandability of imperative process models was investigated in [54]. Efficient cognitive search for UML Class Diagrams [58] and ER Diagrams as well as Data Flow Diagrams was investigated in [26]. Even though all these approaches considerably help to improve the understanding of conceptual models, none of them takes declarative process models into account.

In a similar vein, *guidelines* for creating imperative models, all easier comprehensible, were proposed. For instance, the *Guidelines of Modeling* describe various quality considerations for process models [7]. The so-called *Seven Process Modeling Guidelines* accumulate the insights from various empirical studies (e.g., [36]) to develop a set of actions a system analyst may want to undertake in order to avoid issues with respect to understandability [34]. More generally, guidelines for other conceptual modeling languages, such as ER Diagrams [8], and OWL [45], were proposed as well. Likewise, in [30], the effectiveness and usability of design guidelines for multiple diagrams was evaluated. Though all these guidelines aim to improve the understandability of conceptual models, none of them takes declarative process models into account, as approached in this contribution.

In this work, we have investigated the *understanding* of declarative process models; likewise, the *creation* of declarative process models is closely connected to this work. In particular, the role of understanding declarative process models during *modeling* has been investigated in [67]. Similar to our work, it has been postulated that declarative models are most beneficial when sequential information is directly available, as empirically validated in [63]. In a similar vein, the construction of declarative process models from execution traces is described in [32] and the verification of respective models is discussed in [62]. Even though these works advance the creation of declarative process models, the sense-making of declarative process models is not investigated as detailed as in this contribution. Besides the creation of declarative process models, their *execution*, as enabled by Declare [47], should be mentioned as well. In particular, although declarative process models provide a high degree of flexibility, their execution might pose a significant challenge. As argued in [57], it may not always be clear to end users, which activity shall be executed next. To counterbalance this problem, several methods for guiding the end user through the execution of a declarative process instance were proposed. For instance [47], proposes to generate an automaton which represents all feasible traces related to a declarative process model. From such automation, in turn, optimized execution plans can be created. In a similar way, the approach of [38] can be used to generate and select traces of differing quality factors. Likewise, optimized execution plans may be used for giving recommendations to users [5], generating imperative process models [18], simulation, and time prediction [27]. More broadly [57], propose similar methods that can be applied to imperative process models as well. Even though this approach highlights improving the usability of declarative process models, focus is on the phase of process operation solely.

For this study, we have focused on the declarative modeling language Declare. Recently, also dynamic condition response (DCR) graphs [41] have gained increasing interest. DCR graphs, like Declare, allow for the specification of declarative process models, support the specification of sub-processes [24], and have been applied in a case study for a cross-organizational case management system [25]. Unlike Declare, DCR graphs focus on a set of core constraints instead of allowing for the specification of arbitrary constraints. DCR graphs also employ different formalisms for operationalizing constraints. So far, contributions related to DCR graphs have rather focused on technical aspects, such as technical feasibility, formal correctness, and expressiveness, while understandability issues have not been approached yet.

In this work, we investigated the outcome of a process modeling endeavor, i.e., the process model. Recently, researchers have begun to investigate the process of creating a model, referred to as *the process of process modeling* [50]. Similar to this work, the way in which modelers make sense of a model, while creating it, is investigated, e.g., by visualizing the process of process modeling [11]. Similarly, different personalized modeling styles [49] and modeling strategies have been identified [12]. Even though this stream of research promises to provide insights into the sense-making during modeling, none of these works has investigated the creation of declarative models yet. Rather, current works seem to focus on imperative modeling languages.

6 Summary and outlook

Declarative approaches to business process modeling have recently attracted interest, as they provide a high degree of flexibility [47]. However, the increase in flexibility comes at the cost of understandability and hence might result in maintainability problems of respective process models [47,67]. To advance the understandability of declarative process models, we conducted an empirical investigation, consisting of

an exploratory study and a follow-up study. The exploratory study investigated how system analysts make sense of declarative process models specified in Declare and provided insights into associated problems. Further, the results indicate that system analysts read declarative process models in a sequential way. Regarding the combination of constraints, the exploratory found that single constraints caused only minor problems, but combinations of several constraints seem to be challenging. More specifically, hidden dependencies, as caused by the combination of constraints, were hardly identified.

In the follow-up study, we set out to confirm and refine the findings of the exploratory study. In the exploratory study, we found that system analysts read declarative process models in an iterative and sequential way. However, this finding could not be fully confirmed in the follow-up study. In particular, the follow-up study showed that starting at the init constraint and proceeding in a sequential way, considering the execution order of activities, is a common strategy. However, the analysis also showed that there is a second commonly applied strategy, in which the analysis is started in the top-left corner, reading the model in a top-down manner. In the exploratory study—due to the way models were laid out—these two strategies coincided. Using models with two different layout variants in the follow-up study, we were able to identify these two distinct strategies.

All other findings of the exploratory study could be confirmed and refined in the follow-up study. In particular, the follow-up study showed that single constraints—analogous to the exploratory study—did not cause considerable problems. The combination of constraints, however, caused considerable challenges. These challenges could be observed in the context of several tasks like naming minimal traces, naming valid traces, understanding pairs of constraints, and understanding hidden dependencies. In addition, the study showed that experience regarding declarative process modeling correlates with accuracy, i.e., the amount of correct answers, while experience regarding imperative modeling did not lead to better results. While literature suggests that process modeling knowledge can be transferred from one imperative language to the other quite easily [51], this apparently does not hold for different paradigms. Moreover, regarding the assessment of understandability, a close relationship between mental effort and accuracy could be established. Besides the combination of constraints, aspects of declarative process models that graphically look similar to imperative concepts, e.g., cycle-looking structures and directed edges, were mentioned.

Our studies focused on the sense-making of manually created Declare models, but we did not consider models created with the help of process mining. Even though approaches for mining declarative constraints, e.g., [32], certainly support the creation of declarative process models, it seems likely that certain post-processing steps are necessary. For instance, as the follow-up study showed, a model's layout is an important aspect when trying to make sense of a model. As the entry point of a declarative process model is not necessarily unique [67], automatically laying out a declarative process model may not always be possible. Hence, one option might be to start with a process model that was created by process mining and to manually adapt the model according to the findings of this work, e.g., to clean up the layout and to avoid the usage of paired constraints.

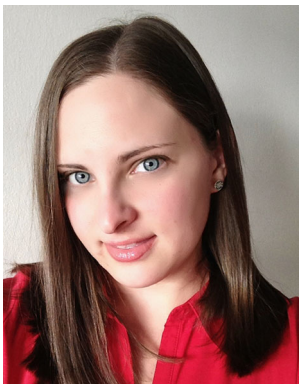
To improve the understandability of declarative process models, it seems particularly promising to make hidden dependencies explicit, e.g., by providing computer-based support, facilitating the interpretation of constraints [67], and to make use of modularization [68]. Regarding designing declarative process modeling notations, it might be recommendable to avoid representing declarative models in a way similar to imperative models, especially when semantic differs considerably (cf. Mindshift Learning theory [3]). Guidelines aiming to improve the understandability of conceptual models are introduced in Sect. 5, but guidelines easing the sense-making of declarative process model in order to avoid pitfalls are still missing. Sections 3.3 and 4.3 show frequently occurring difficulties (e.g., paired constraints and hidden dependencies) that should be specifically addressed in class. Even though the data provided first insights into the process of understanding declarative models, further investigations are needed. Particularly, replications utilizing more complex models seem to be appropriate means for additional empirical tests. Although the think-aloud protocols already provided a detailed view on the reasoning processes of system analysts, we plan to employ eye movement analysis for more detailed analysis. The adoption of eye tracking, as discussed in [48], allows for accurately identifying parts of the process model system analysts are focusing on. In addition, this technology allows for constantly assessing the mental effort [23], allowing for the identification of particularly challenging parts in the process model. Based on these insights, we intend to evolve our work toward empirically founded guidelines enabling better understandability of declarative process models.

References

1. Aalst, W., Pesic, M.: Decserflow: towards a truly declarative service flow language. In Proceedings of WS-FM'06, pp. 1–23 (2006)

2. Aranda, J., Ernst, N., Horkoff, J., Easterbrook, S.: A framework for empirical evaluation of model comprehensibility. In: Proceedings of MISE'07, pp. 7–12 (2007)
3. Armstrong, D.J., Hardgrave, B.C.: Understanding mindshift learning: the transition to object-oriented development. MIS Q. **31**(3), 453–474 (2007)
4. Baddeley, A.: Working memory: theories, models, and controversies. Annu. Rev. Psychol. **63**(1), 1–29 (2012)
5. Barba, I., Weber, B., Valle, C.D., Ramírez, A.J.: User recommendations for the optimized execution of business processes. Data Knowl. Eng. **86**, 61–84 (2013)
6. Bassey, M.: Case Study Research in Educational Settings. Doing Qualitative Research in Educational Settings. Open University Press, New York (1999)
7. Becker, J., Rosemann, M., Uthmann, C.: Guidelines of business process modeling. In: Business Process Management, Models, Techniques, and Empirical Studies, pp. 30–49, London, UK. Springer (2000)
8. Bodart, F., Patel, A., Sim, M., Weber, R.: Should optional properties be used in conceptual modelling? A theory and three empirical tests. Inf. Syst. Res. **12**(4), 384–405 (2001)
9. Burkhardt, J.-M., Détienne, F., Wiedenbeck, S.: Object-oriented program comprehension: effect of expertise, task and phase. Empir. Softw. Eng. **7**(2), 115–156 (2002)
10. Burton-Jones, A., Meso, P.N.: Conceptualizing systems for understanding: an empirical test of decomposition principles in object-oriented analysis. Inf. Syst. Res. **17**(1), 38–60 (2006)
11. Claes, J., Vanderfeesten, I., Pinggera, J., Reijers, H., Weber, B., Poels, G.: Visualizing the process of process modeling with PPM-Charts. In: Proceedings of TAProViz '12, pp. 744–755 (2013)
12. Claes, J., Vanderfeesten, I., Reijers, H., Pinggera, J., Weidlich, M., Zugal, S., Fahland, D., Weber, B., Mendling, J., Poels, G.: Tying process model quality to the modeling process: the impact of structuring, movement, and speed. In: Proceedings of BPM'12, pp. 33–48 (2012)
13. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage, London (2007)
14. Costain, G.F.: Cognitive support during object-oriented software development: the case of UML diagrams. Ph.D. thesis, University of Auckland (2007)
15. Cowan, N.: The magical number 4 in short-term memory: a reconsideration of mental storage capacity. Behav. Brain Sci. **24**(1), 87–185 (2001)
16. Ericsson, K.A., Simon, H.A.: Protocol Analysis: Verbal Reports as Data. MIT Press, Cambridge (1993)
17. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: the issue of understandability. In: Proceedings of EMMSAD'09, pp. 353–366 (2009)
18. Ferreira, H., Ferreira, D.: An integrated life cycle for workflow management based on learning and planning. Int. J. Coop. Inf. Syst. **15**(4), 485–505 (2006)
19. Goldberg, J.H., Stimson, M.J., Lewenstein, M., Scott, N., Wichansky, A.M.: Eye Tracking in web search tasks: design implications. In: Proceedings of ETRA'02, pp. 51–58 (2002)
20. Green, T.R., Petre, M.: Usability analysis of visual programming environments: a 'cognitive dimensions' framework. J. Vis. Lang. Comput. **7**(2), 131–174 (1996)
21. Haisjackl, C.: Test driven modeling meets declarative process modeling: a case study. Master's thesis, University of Innsbruck, Aug 2012
22. Haisjackl, C., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Making sense of declarative process models: common strategies and typical pitfalls. In: Proceedings of BPMDS'13, pp. 2–17 (2013)
23. Heitz, R.P., Schrock, J.C., Payne, T.W., Engle, R.W.: Effects of incentive on working memory capacity: behavioral and pupillometric data. Psychophysiology **45**(1), 119–129 (2008)
24. Hildebrandt, T., Mukkamala, R., Slaats T.: Nested dynamic condition response graphs. In: Proceedings of FSEN'11, pp. 343–350 (2012)
25. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Designing a cross-organizational case management system using dynamic condition response graphs. In: Proceedings of EDOC'11, pp. 161–170 (2011)
26. Hungerford, B.C., Hevner, A.R., Collins, R.W.: Reviewing software diagrams: a cognitive study. IEEE Trans. Software Eng. **30**(2), 82–96 (2004)
27. Jiménez-Ramírez, A., Barba, I., Del Valle, C., Weber, B.: Generating multi-objective optimized business process enactment plans. In: Proceedings of CAiSE'13, pp. 99–115 (2013)
28. Kahneman, D.: Maps of bounded rationality: a perspective on intuitive judgment and choice. Nobel Prize Lect. **8**, 449–489 (2002)
29. Khatri, V., Vessey, I., Ramesh, P.C.V., Park, S.-J.: Understanding conceptual schemas: exploring the role of application and IS domain knowledge. Inf. Syst. Res. **17**(1), 81–99 (2006)
30. Kim, J., Hahn, J., Hahn, H.: How do we understand a system with (so) many diagrams? Cognitive integration processes in diagrammatic reasoning. Inf. Syst. Res. **11**(3), 284–303 (2000)
31. Kim, Y.-G., March, S.T.: Comparing data modeling formalisms. Commun. ACM **38**(6), 103–115 (1995)
32. Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Inducing declarative logic-based models from labeled traces. In: Proceedings of BPM'07, pp. 344–359 (2007)
33. Mayer, R., Chandler, P.: When learning is just a click away: does simple user interaction foster deeper understanding of multimedia messages. J. Educ. Psychol. **93**(2), 390–397 (2001)
34. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Inf. Softw. Technol. **52**(2), 127–136 (2010)
35. Mendling, J., Strembeck, M., Recker, J.: Factors of process model comprehension—findings from a series of experiments. Decis. Support Syst. **53**(1), 195–206 (2012)
36. Mendling, J., Verbeek, H., van Dongen, B., van der Aalst, W., Neumann, G.: Detection and prediction of errors in EPCS of the sap reference model. Data Knowl. Eng. **64**(1), 312–329 (2008)
37. Miller, G.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956)
38. Montali, M.: Specification and verification of declarative open interaction models: a logic-based approach. Ph.D. thesis, Department of Electronics, Computer Science and Telecommunications Engineering, University of Bologna (2009)
39. Montali, M., Pesic, M., van der Aalst, W., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographies. ACM Trans. Web **4**(1), 1–62 (2010)
40. Moody, D.L.: Cognitive load effects on end user understanding of conceptual models: an experimental analysis. In: Proceedings of ADBIS'04, pp. 129–143 (2004)
41. Mukkamala, R.R.: A formal model for declarative workflows-dynamic condition response graphs. Ph.D. thesis, IT University of Copenhagen (2012)
42. Mylopoulos, J.: Information modeling in the time of the revolution. Inf. Syst. **23**(3/4), 127–155 (1998)
43. Nielsen, J.: Estimating the number of subjects needed for a thinking aloud test. Int. J. Hum.–Comput. Stud. **41**(3), 385–397 (1994)
44. Paas, F., Renkl, A., Sweller, J.: Cognitive load theory and instructional design: recent developments. Educ. Psychol. **38**(1), 1–4 (2003)
45. Palash, B., Burton-Jones, A., Wand, Y.: Guidelines for designing visual ontologies to support knowledge identification. MIS Q. **35**(4), 883–908 (2011)

46. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15**(12), 1053–1058 (1972)
47. Pesic, M.: Constraint-based workflow management systems: shifting control to users. Ph.D. thesis, TU Eindhoven (2008)
48. Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., Weber, B.: Investigating the process of process modeling with eye movement analysis. In: Proceedings of ER-BPM'12, pp. 438–450 (2013)
49. Pinggera, J., Soffer, P., Fahland, D., Weidlich, M., Zugal, S., Weber, B., Reijers, H., Mendling, J.: Styles in business process modeling: an exploration and a model. Softw. Syst. Model. (2013). doi:10.1007/s10270-013-0349-1
50. Pinggera, J., Zugal, S., Weidlich, M., Fahland, D., Weber, B., Mendling, J., Reijers, H.: Tracing the process of process modeling with modeling phase diagrams. In: Proceedings of ER-BPM'11, pp. 370–382 (2012)
51. Recker, J.C., Dreiling, A.: Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education. In: Proceedings of ACIS'07, pp. 356–366 (2007)
52. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Berlin (2012)
53. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Syst. **36**(5), 881–897 (2011)
54. Reijers, H.A., Mendling, J.: A study into the factors that influence the understandability of business process models. IEEE Trans. Syst. Man Cybern. A **41**(3), 449–462 (2011)
55. Reijers, H.A., Slaats, T., Stahl, C.: Declarative modeling—an academic dream or the future for BPM? In: Proceedings of BPM'13, pp. 307–322 (2013)
56. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int. J. Hum.–Comput. Stud. **45**(2), 185–213 (1996)
57. Schonenberg, H., Weber, B., van Dongen, B., van der Aalst, W.M.P.: Supporting flexible processes through recommendations based on history. In: Proceedings of BPM'08, pp. 51–66 (2008)
58. Shehnaaz, Y., Kagdi, H., Maletic, J.: Assessing the comprehension of UML class diagrams via eye tracking. In: Proceedings of ICPC'07, pp. 113–122 (2007)
59. Strauss, A., Corbin, J.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage, London (1998)
60. Sweller, J.: Cognitive load during problem solving: effects on learning. Cogn. Sci. **12**(2), 257–285 (1988)
61. Unsworth, N., Engle, R.W.: The nature of individual differences in working memory capacity: active maintenance in primary memory and controlled search from secondary memory. Psychol. Rev. **114**(1), 104–132 (2007)
62. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.: Process mining and verification of properties: an approach based on temporal logic. In: Proceedings of OTM'05, pp. 130–147 (2005)
63. Zugal, S., Haisjackl, C., Pinggera, J., Weber, B.: Empirical evaluation of test driven modeling. Int. J. Inf. Syst. Model. Des. **4**(2), 23–43 (2013)
64. Zugal, S., Pinggera, J., Mendling, J., Reijers, H., Weber, B.: Assessing the impact of hierarchy on model understandability—a cognitive perspective. In: Proceedings of EESSMod'11, pp. 123–133 (2011)
65. Zugal, S., Pinggera, J., Reijers, H., Reichert, M., Weber, B.: Making the case for measuring mental effort. In: Proceedings of EESSMod'12, pp. 37–42 (2012)
66. Zugal, S., Pinggera, J., Weber B.: Assessing process models with cognitive psychology. In: Proceedings of EMISA'11, pp. 177–182 (2011)
67. Zugal, S., Pinggera, J., Weber, B.: Toward enhanced life-cycle support for declarative processes. J. Softw. Evol. Process **24**(3), 285–302 (2012)
68. Zugal, S., Soffer, P., Haisjackl, C., Pinggera, J., Reichert, M., Weber, B.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Softw. Syst. Model. (2013). doi:10.1007/10270-013-0356-2



Cornelia Haisjackl is a Ph.D. candidate at the University of Innsbruck (Austria). She is a member of Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. Cornelia received her M.Sc. degree from the Department of Computer Science, University of Innsbruck in 2012. Her main research interests are declarative business process models and the understandability of business process models.

Irene Barba is a lecturer in Computer Science at the University of Seville, Spain. She holds M.Sc. and Ph.D. degrees in Computer Science at the University of Seville. Irene's research is focused on the application of artificial intelligence techniques to enhance the effectiveness of the different phases of the business process management life cycle. Specifically, her research interests include the optimized management of business processes, user support in flexible process-aware information systems, constraint programming, planning, and scheduling. She published papers at international conferences and journals (e.g., Data and Knowledge Engineering, Information and Software Technology, Journal of Intelligent Manufacturing, International Journal of Cooperative Information Systems, CAiSE). She is a member of the Quivir Research Group at the University of Seville.

Stefan Zugal is a postdoctoral researcher at the University of Innsbruck (Austria), where he received his doctorate at the Department of Computer Science in 2014. Stefan is member of the Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. His main research interests are declarative business process models and the understandability of business process models. Stefan has published more than 40 refereed papers in international journals, conferences, and workshops.

Pnina Soffer is a senior lecturer in the Information Systems Department at the University of Haifa. She received her B.Sc. (1991) and M.Sc. (1993) in Industrial Engineering, Ph.D. in Information Systems Engineering from the Technion, Israel Institute of Technology (2002). Her research deals with business process modelling and management, requirements engineering, and conceptual modelling, addressing issues such as goal orientation, flexibility, interoperability, and context-aware adaptation. She has published over 90 papers in journals and conference proceedings. Among others, her research has appeared in journals such as Journal of the AIS, European J of IS, Requirements Engineering, Information Systems, and more. She has served as a guest editor for a number of special issues related to various business process topics such as business process flexibility, coordinated development of business processes and information systems, and business process design. Pnina has served in program committees of numerous conferences, including CAiSE, BPM, ER, and held several organizational positions in CAiSE and in BPM, including program chair of BPM 2014. She is a member of the CAiSE steering committee and an organizer of the BPMDS working conference. She is a member of editorial boards of several journals including the Journal of the AIS.

Irit Hadar is a tenured faculty member at the Department of Information Systems at the University of Haifa. She is the Head of the Software Architecture Laboratory at the Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science. Her main research area is cognitive aspects of software architecture, design, and analysis. She published over 70 papers in international journals and conferences (CACM, JAIS, EJIS, REJ, IST, JSS, JISE, OOPSLA, AMCIS, MCIS, etc.). She has been serving as an organizer and PC member in conferences and workshops (CAiSE, ICIS, AMCIS, MCIS, OOPSLA), and as an editorial board member (ACM TOCE journal). She holds a Ph.D. from the Technion, Israel Institute of Technology.

Manfred Reichert is Professor at the University of Ulm, Germany, and Director of the Databases and Information Systems Institute. His major research interests include business process management (e.g., technologies supporting process flexibility, data-centric processes, knowledge-intensive processes, and mobile processes), service-oriented computing (e.g., service composition, service evolution), and process-aware information systems in advanced application domains (e.g., e-health and automotive engineering). Manfred pioneered the work on the ADEPT process management technology and is co-founder of the AristaFlow GmbH. He has been participating in numerous research projects in the BPM area contributing more than 250 scientific papers on BPM-related topics. His book entitled "Enabling Flexibility in Process-Aware Information Systems," which he co-authored with Barbara Weber, was published by Springer in September 2012. Manfred has been PC Co-Chair of the BPM'08, CoopIS'11, and EDOC'13 conferences and General Chair of the BPM'09 and EDOC'14 conferences.

Jakob Pinggera is a Ph.D. candidate at the University of Innsbruck (Austria). Jakob is a member of Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. Jakob received his M.Sc. degree from the Department of Computer Science, University of Innsbruck in 2009. His main research interest is the process of process modeling, i.e., how modelers create business process models. His research interests further include business process modeling notations and business process quality. Jakob has published more than 40 refereed papers in international journals, conferences, and workshops.

Barbara Weber is Associate Professor at the University of Innsbruck (Austria). Barbara is a member of the Quality Engineering (QE) Research Group and Head of the Research Cluster on Business Processes and Workflows at QE. Barbara holds a Habilitation degree in Computer Science and Ph.D. from the University of Innsbruck. Barbara's research interests include process model understandability, process of process modeling, integrated process life cycle support, change patterns, process flexibility, user support in flexible process-aware systems, and recommendations to optimize process execution. Barbara has published more than 130 refereed papers, for example, in Data and Knowledge Engineering and Software and System Modeling, Computers in Industry, Science of Computer Programming, Enterprise Information Systems and Information and Software Technology and co-author of the book "Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies" by Springer. Moreover, Barbara the organizer of the successful BPI workshop series has been PC Co-Chair of BPM 2013 and will be general chair of BPM 2015 in Innsbruck.