How Understandable Are Pattern-Based Behavioral Constraints for Novice Software Designers?

CHRISTOPH CZEPA, University of Vienna, Austria
UWE ZDUN, University of Vienna, Austria

This article reports a controlled experiment with 116 participants on the understandability of representative graphical and textual pattern-based behavioral constraint representations from the viewpoint of novice software designers. Particularly, graphical and textual behavioral constraint patterns present in the declarative business process language *Declare* and textual behavioral constraints based on *Property Specification Patterns* are the subjects of this study. In addition to measuring the understandability construct, this study assesses subjective aspects such as perceived difficulties regarding learning and application of the tested approaches. An interesting finding of this study is the overall low achieved correctness in the experimental tasks which seems to indicate that pattern-based behavioral constraint representations are hard to understand for novice software designers in the absence of additional supportive measures. The results of the descriptive statistics regarding achieved correctness are slightly in favor of the textual representations, but the inference statistics do not indicate any significant differences in terms of understandability between graphical and textual behavioral constraint representations.

CCS Concepts: • **Software and its engineering → Formal language definitions**.

Additional Key Words and Phrases: Controlled experiment, understandability, behavioral constraints, property specification patterns, declarative business processes

ACM Reference Format:
Christoph Czepa and Uwe Zdun. 2019. How Understandable Are Pattern-Based Behavioral Constraints for Novice Software Designers?. *ACM Trans. Softw. Eng. Methodol.* 1, 1, Article 1 (January 2019), 39 pages.
https://doi.org/10.1145/3306608

1 INTRODUCTION

Since the early days of computer science, supporting the correctness of computer programs has been a recurring research interest. In 1977, Pnueli introduced an approach for the verification of sequential and parallel programs that is based on temporal reasoning [65]. His approach became widely popular under the term *Linear Temporal Logic (LTL)*. A plethora of different temporal logics have been proposed since then. For example, in 1988, Clarke and Emerson [10] applied the *Computation Tree Logic (CTL)*, a branching time logic, for model checking of computer programs. Both LTL and CTL are still very popular and supported as a specification language by many of today's model checkers (e.g., NuSMV by Cimatti et al. [9] and SPIN by Holzmann [39]).[1]

[1] http://nusmv.fbk.eu, http://spinroot.com

Authors' addresses: Christoph Czepa, University of Vienna, Währingerstraße 29, Vienna, Vienna, 1190, Austria, christoph.czepa@univie.ac.at; Uwe Zdun, University of Vienna, Währingerstraße 29, Vienna, Vienna, 1190, Austria, uwe.zdun@univie.ac.at.

1049-331X/2019/1-ART1 $15.00
https://doi.org/10.1145/3306608

In 1998, Dwyer et al. [23, 24] proposed the *Property Specification Patterns (PSP)*, a pattern-based approach that abstracts underlying LTL, CTL, or other formal logic formulas.[2] As the main reason for collecting the patterns, the authors state that practitioners had been reluctant to apply formal methods due to unfamiliarity with specification processes, notations, and strategies. Consequently, the pattern catalog was primarily meant to enable ease of reuse of existing patterns. The proposed set of patterns was evaluated against 555 specifications from more than 35 different sources, and 92.1% (511) of the considered specifications are covered by the proposed set of patterns. A survey by Bianculli et al. [7] based on 104 scientific case studies reproduced these results.

Numerous studies make use of the PSP approach either directly or by extending the original idea of pattern-based behavioral constraints. In the following, we will discuss a selection of them to emphasize the importance and applicability of PSP for various purposes and to introduce the *Declare* approach (cf. Pešić et al. [61]), a popular graphical behavioral constraint approach, which has been greatly inspired by PSP.

Corbett et al. [12] apply PSP for the verification of Java programs. PSP can be used for modeling requirements in requirements engineering (see e.g., Cheng & Atlee [8]). Hatcliff et al. [35] apply PSP for the verification of component-based systems. Krismayer et al. [48] propose an approach that mines constraints from event logs on the basis of PSP. Li et al. [49] use a structured textual specification language based on PSP for the behavioral verification of webservices at runtime. Wong and Gibbons [85] apply a superset of PSP to construct behavioral properties of BPMN (Business Process Model and Notation) [59] models. Namiri and Stojanovic [58] propose a PSP-based approach for modeling internal controls that are required by regulations (e.g., Sarbanes-Oxley Act of 2002) for business process compliance. Elgammal et al. [25] adopted some of the *Property Specification Patterns* in the *Compliance Request Language (CRL)*. Dou et al. [22] extended the Object Constraint Language (OCL) with support for temporal constraints based on PSP. The *PROPEL (PROPerty Elicitation)* approach by Smith et al. [76] provides support for the specification of PSP-based constraints using two different notations, namely an extended finite-state automaton representation and a structured natural language representation. The *PROPOLS* approach for the verification of BPEL service compositions schemas (cf. Yu et al. [86]) is based on PSP as well. Awad et al. [1] propose a PSP-based visual language, called *BPMN-Q*, to express compliance requirements with visual shapes that are similar to those used in imperative business process modeling.

Both the graphical declarative workflow approach *Declare* (formerly known as *ConDec*) by Pešić et al. [61] and the graphical *DecSerFlow (Declarative Service Flow Language)* approach by van der Aalst and Pešić [80] were strongly inspired by the PSP approach.[3] *Declare* appears to be the most wide-spread graphical behavioral constraint approach in business process management (cf. Goedertier et al. [32], Schonenberg et al. [73], and van der Aalst et al. [81]), and its abstractions are generic constraint language abstractions. That is, it can be seen as representative for, and generalized to, a broad set of possible other graphical behavioral constraint languages.

1.1 Problem Statement

Behavioral constraint approaches are highly relevant in various domains, such as health care (cf. Rovani et al. [71]), banking (cf. Bianculli et al. [7]), automotive (cf. Post et al. [66]), software architecture (cf. Czepa et al. [14]), and business process management (cf. Elgammal et al. [25]), to name but a few. Pattern-based behavioral constraints can be used to shield the user from the complexity of formal temporal logics used in the context of formal verification methods such as model checking (cf. Rozier [72] and Baier & Katoen [2]) and runtime monitoring by nondeterministic

[2] http://patterns.projects.cs.ksu.edu
[3] http://www.win.tue.nl/declare/2011/11/declare-renaming/

finite automata (cf. De Giacomo et al. [19, 20]). Many textual and graphical pattern-based behavioral constraint approaches exist (e.g., [1, 25, 80, 86]) which originated from PSP. However, current studies predominantly focus on technical contributions in specific application areas. Only a few studies focus on empirical evaluations of behavioral constraint representations [14, 21, 34, 64, 82, 87], and even fewer of them are concerned with comparing graphical and textual behavioral constraint representations specifically [33, 52]. Interestingly, the body of existing studies (cf. Section 8 which discusses those related works in depth) yields contradictory results which indicates that the understandability of pattern-based graphical and textual behavioral constraints is not yet well understood. Two prior empirical studies (both reported in [16]) indicated that the pattern-based PSP representation provides a high level of understandability (about 70 percent on average in the specific setup of those studies), but these studies did not consider graphical pattern-based behavioral constraints. As it is our experience from multiple industry projects that industry experts in areas such as business process management tend to prefer graphical over textual constraint representations when given the choice, it would be important to test if their gut feeling can be empirically confirmed. Also, non-expert users seem to prefer graphical models over structured text and textual descriptions when the goal is to understand a process (cf. Figl & Recker [30]). It is yet unknown whether there are differences in understandability between graphical and textual pattern-based behavioral constraint approaches. In addition, it is unknown whether there exist problematic language elements that pose an obstacle for correct understanding of textual and graphical pattern-based behavioral constraint representations. The discovery of such problematic elements could provide a starting point for improving the comprehensibility of the representations and making them more applicable in practice.

Studying the understandability of graphical and textual behavioral constraints is not only interesting from a purely scientific point of view, but it is also important for industrial applications. For example, from the cooperation with our industry partners (see e.g., [79]), their customers, and other company representatives at conferences and workshops, we realized that the industry has a huge demand for, and shows a strong interest in, behavioral constraint approaches that are applicable in practice by supporting the comprehensible, fast and accurate adoption of compliance requirements, as well as their automated enactment and verification. The pattern-based behavioral constraint representations that we study in this article are well-suited for automated computer-aided verification at runtime and design time, but vendors are still often reluctant to expose their customers to such approaches. Our discussions with industry partners (see e.g. [77, 78]) indicate uncertainty regarding how understandable the constraints are, and this could be among the reasons for this reluctance.

In addition to triggering further empirical evaluations and thus new insights, empirical research on behavioral constraints has the potential to influence practitioners in decision-making for adopting a specific behavioral constraint language and in designing future industrial solutions. Consequently, a farther-reaching goal for our research on behavioral constraint representations is to pave the way for their future industrial or practical exploitation.

1.2 Research Objectives

This empirical study has the objective to investigate the understandability of representative graphical and textual behavioral constraint representations. The understandability construct focuses on how well (in terms of correct understanding) and fast (in terms of the response time) a participant understands a given behavioral constraint representation. Particularly, this empirical study considers the *Property Specification Patterns*, which are the origin of numerous existing behavioral constraint approaches (cf. Section 1), and the *Declare* approach, which seems to be the most popular graphical behavioral constraint approach in the field of business process management (cf.

Goedertier et al. [32], Schonenberg et al. [73], and van der Aalst et al. [81]). We are not aware of any other graphical behavioral constraint language being of similar relevance. Originally, *Declare* was proposed in the domain of business process management (cf. Pešić & van der Aalst [62]) and also applied in service-oriented computing (cf. van der Aalst & Pešić [80]), but there seem to be no limiting factors for the application of *Declare* in different domains. Its graphical pattern-based representation is versatile and transformable to underlying formal representations (e.g., LTL [54] and event calculus [55]) for verification at design time (i.e., model checking) and runtime verification in general. *Declare* is considered in three variants, namely as a purely graphical, a purely textual, and a hybrid (mixed graphical/textual) behavioral constraint approach.

We state the experimental goal using the GQM (Goal Question Metric) goal template (cf. Basili et al. [3]) as follows:

Analyze the textual Property Specifications Patterns (PSP) based representation approach, the purely graphical *Declare* representation approach (DG), the purely textual *Declare* representation approach (DT), and the hybrid (i.e., showing a textual label in addition to the graphical relation) *Declare* representation approach (DGT)
for the purpose of their evaluation
with respect to their understandability
from the viewpoint of the novice software designer
in the context (i.e., environment) of the Distributed System Engineering and the Software Engineering 2 courses at the University of Vienna, Austria.

1.3 Guidelines

Jedlitschka et al. [42] propose guidelines for reporting experiments, which had a strong influence on the general structure of this article. Those guidelines integrate (among others) the "Preliminary guidelines for empirical research in software engineering" by Kitchenham et al. [46] and standard books on empirical software engineering (cf. Wohlin et al. [84], Juristo & Moreno [44]). Moreover, the "Robust Statistical Methods for Empirical Software Engineering" by Kitchenham et al. [45] had a strong impact on the statistical methods used for the evaluation of the gathered data.

2 BACKGROUND ON PATTERN-BASED BEHAVIORAL CONSTRAINT REPRESENTATIONS

2.1 Property Specification Patterns

Dwyer et al. [23, 24] proposed the *Property Specification Patterns (PSP)*, a collection of recurring behavioral constraint patterns. Since the patterns cannot be directly used for formal verification, there exist transformations to underlying formal representations (among them are *Linear Temporal Logic (LTL)* [65] and *Computation Tree Logic (CTL)* [10] formulas) that can be found online.[4] The discovered patterns were categorized into *Occurrence Patterns* and *Order Patterns*:

- Occurrence Patterns:
  - Absence: *a* `never occurs`
  - Universality: *a* `always occurs`
  - Existence: *a* `occurs`
  - Bounded Existence: *a* `occurs at most` *n* `times`
- Order Patterns:
  - Precedence: *a* `precedes` *b*
  - Response: *a* `leads to` *b*

[4] http://patterns.projects.cs.ksu.edu/documentation/patterns.shtml

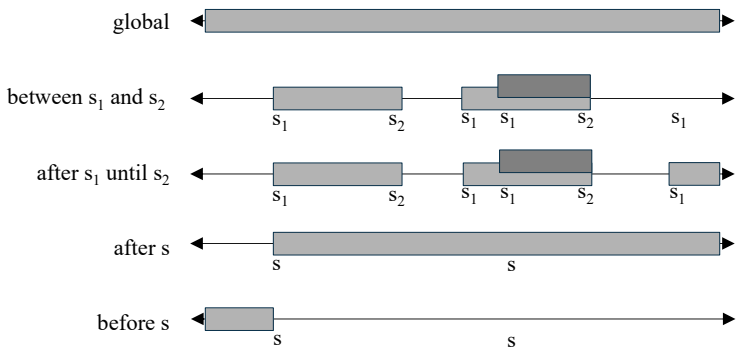


Fig. 1. Available scopes for *Property Specification Patterns* (shaded areas indicate the extent over which the pattern must hold)

  - 2 Cause-1 Effect Precedence Chain: `(`a`, `b`) precedes `c
  - 1 Cause-2 Effect Precedence Chain: a `precedes (`b`, `c`)`
  - 2 Stimulus-1 Response Chain: `(`a`, `b`) leads to `c
  - 1 Stimulus-2 Response Chain: a `leads to (`b`, `c`)`

Each pattern has a *scope*. Figure 1 shows the available scopes and their area of effect:

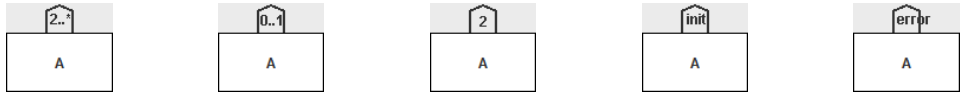
- The *global* scope defines that a pattern must hold during the entire execution of a system. If no scope is defined, this scope is implicitly assumed.
- The *before* scope `before` s `[` p `]` defines that a pattern p must hold before the first occurrence of s.
- The *after* scope `after` s `[` p `]` defines that a pattern p must hold after the first occurrence of s.
- The *between* scope `between` s_1 `and` s_2 `[` p `]` defines that a pattern p must hold between every s_1 (i.e., starting the scope) that is followed by s_2 (i.e., closing the scope).
- The *after-until* scope `after` s_1 `until` s_2 `[` p `]` defines that a pattern p must hold after every s_1 (i.e., starting the scope) by no later than s_2 (i.e., closing the scope).

2.2 Declare

Declare (cf. Pešić & van der Aalst [61]), also known by the names *DecSerFlow* (cf. van der Aalst & Pešić [80]) and *ConDec* (cf. Pešić & van der Aalst [62]), is a graphical declarative business process modeling language and approach. There exist transformations of its high-level graphical representations to *Linear Temporal Logic (LTL)* (cf. Pnueli [65] and Montali [54]) and *Event Calculus (EC)* (cf. Kowalski & Sergot [47] and Montali et al. [55]). As of *Declare Version 2.1.0*, the available constraint templates are organized as follows:[5]

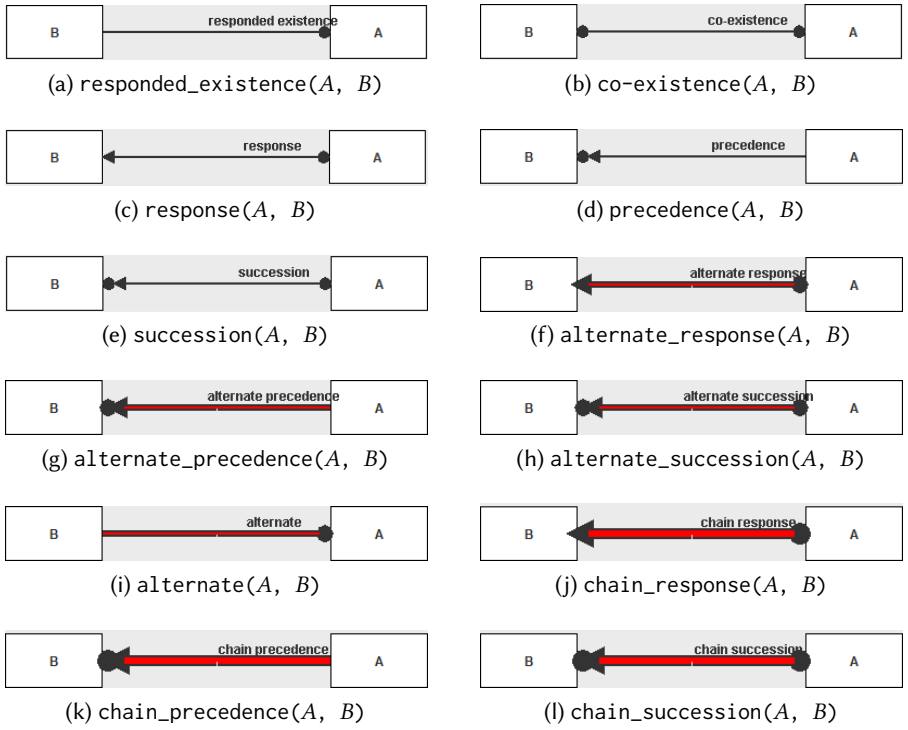
- Existence Patterns (cf. Figure 2 for graphical representations):
  - "at least"
    - `existence_`n`(`A`)`: State A must occur at least n times.
  - "at most"
    - `absence_`n`(`A`)`: State A must occur at most $n - 1$ times.
  - "exactly"
    - `exactly_`n`(`A`)`: State A must occur exactly n times (i.e., not more, not less).
  - "position"

[5] http://www.win.tue.nl/declare/download/



(a) `existence_2(`A`)` (b) `absence_2(`A`)` (c) `exactly_2(`A`)` (d) `init(`A`)` (e) `error(`A`)`

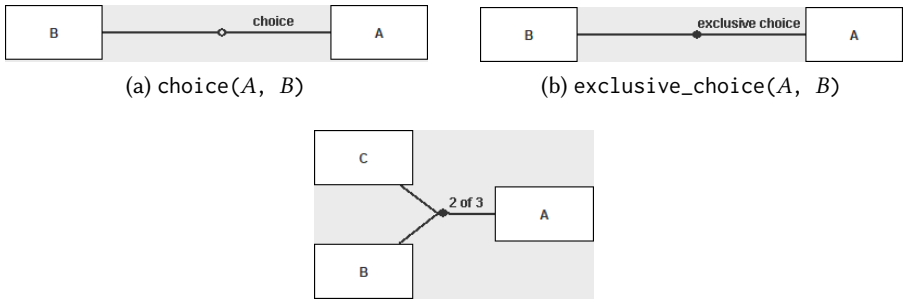
Fig. 2. Graphical representations of *Existence* patterns in Declare



(a) `responded_existence(`A`, `B`)` (b) `co-existence(`A`, `B`)`
(c) `response(`A`, `B`)` (d) `precedence(`A`, `B`)`
(e) `succession(`A`, `B`)` (f) `alternate_response(`A`, `B`)`
(g) `alternate_precedence(`A`, `B`)` (h) `alternate_succession(`A`, `B`)`
(i) `alternate(`A`, `B`)` (j) `chain_response(`A`, `B`)`
(k) `chain_precedence(`A`, `B`)` (l) `chain_succession(`A`, `B`)`

Fig. 3. Graphical representations of *Relation* patterns in Declare

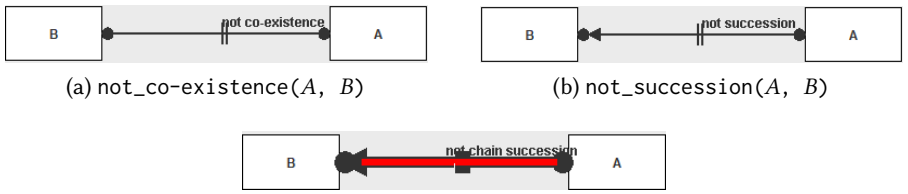
    - `strong_init(`A`)`: A must start and complete first.
    - `init(`A`)`: A must start first, and it must complete first or remain active indefinitely.
    - `last(`A`)`: A must be the last occurring element. There must not occur any other element than A after A.
  - `error(`A`)`: This appears to be an auxiliary pattern to detect a completion of A that should not occur if A has never been started.
- Relation Patterns (cf. Figure 3 for graphical representations):
  - "no order"
    - `responded_existence(`A`, `B`)`: If state A happens (at least once), then state B must have happened (at least once) before state A or must happen after state A.
    - `co-existence(`A`, `B`)`: If state A happens (at least once), then state B must have happened (at least once) before state A or must happen after state A, and vice versa.



(a) choice(A, B) (b) exclusive_choice(A, B)

(c) exclusive_choice_2_of_3(A, B, C)

Fig. 4. Graphical representations of *Choice* patterns in Declare



(a) not_co-existence(A, B) (b) not_succession(A, B)

(c) not_chain_succession(A, B)

Fig. 5. Graphical representations of *Negative Relation* patterns in Declare

– "order"
  * "simple"
    · response(A, B): Whenever state A happens, state B must occur afterwards eventually.
    · precedence(A, B): The occurrence of state A is a precondition for state B. State B is only allowed to happen if state A has happened already.
    · succession(A, B): Whenever state A happens, state B must occur afterwards eventually. The occurrence of state A is a precondition for state B. State B is only allowed to happen if state A has happened already. That is, this pattern is a combination of response and precedence.
  * "alternate"
    · alternate_response(A, B): Whenever state A happens, state B must occur afterwards eventually, but A is not allowed to occur a second time until then.
    · alternate_precedence(A, B): A must occur before the first B, then the occurrence of another A is the precondition for the next B, and so forth.
    · alternate_succession(A, B): This pattern is a combination of alternate_response and alternate_precedence.
    · alternate(A, B): After A there must not be another A indefinitely or until B occurs
  * "chain"
    · chain_response(A, B): Whenever state A happens, state B must occur next.
    · chain_precedence(A, B): B can only be executed directly after A.

    - chain_succession(A, B): This pattern is a combination of chain_response and chain_precedence.
- Choice Patterns (cf. Figure 4 for graphical representations):
  - "simple"
    - choice(A, B): State A or state B must occur. That is, either of them occurring alone would satisfy this constraint, but both may occur anyway. For example, the traces [A, B], [A], [B] would satisfy this constraint while an empty trace [] would cause a violation.
    - choice_n_of_N(list[N]): This pattern is the generalization of choice where $n, N \in \mathbb{N}$, $N \geq 2$, and $1 \leq n < N$. For example, the traces [A, B, C], [A, B], [A, C], [B, C] would satisfy the constraint choice_2_of_3(A, B, C) while the traces [], [A], [B], [C] would cause violations.
  - "exclusive"
    - exclusive_choice(A, B): State A or state B must occur, but not both. That is, either of them occurring alone would satisfy this constraint, and the constraint would be violated if both of them occur. For example, the traces [A], [B] would satisfy this constraint while the traces [], [A, B] would cause violations.
    - exclusive_choice_n_of_N(list[N]): Generalization of exclusive_choice where $n, N \in \mathbb{N}$, $N \geq 2$, and $1 \leq n < N$. For example, the traces [A, B], [A, C], [B, C] would satisfy the constraint exclusive_choice_2_of_3(A, B, C) while the traces [], [A], [B], [C], [A, B, C] would cause violations.
- Negative Relation Patterns (cf. Figure 5 for graphical representations):
  - "no order"
    - not_co-existence(A, B): Either state A or state B can occur, but not both.
  - "order"
    - not_succession(A, B): Before state B there cannot be state A and after state A there cannot be state B.
  - "chain"
    - not_chain_succession(A, B): A and B must not occur next to each other in this order.

3 EXPERIMENT PLANNING

3.1 Goals

The primary goal of the experiment is measuring the construct *understandability* of graphical and textual pattern-based behavioral constraint representations by the *correctness* and *response time* of the answers given by the participants.

Additionally, the experiment aims at studying the perceived learning difficulty, the perceived difficulty regarding applying the learned behavioral constraint representation approach (i.e., the perceived application difficulty), the personal interest in using the representation, the perceived practical applicability, and the perceived potential for further improvement of the behavioral constraint representations.

3.2 Experimental Units

All 116 participants were students at the University of Vienna, Austria, who enrolled in the courses "Distributed System Engineering Lab (DSE)" and "Software Engineering 2 (SE2)" in the winter term 2017. This study aims at evaluating the understandability of pattern-based behavioral constraints from the perspective of novice software designers, which makes undergraduate students suitable test subjects. The attendance was optional and rewarded by extra credits (i.e., bonus points) for the course based on the performance in the experiment (i.e., the achieved correctness and

completeness of time records). Alternatively, the students were given the chance to gain extra credits in other lab activities by going beyond the normal course requirements (e.g., by implementing more functionalities than required, or paying attention to excellent code quality). As required for a valid controlled experiment setup, all participants were randomly allocated to the four experiment groups (i.e., one for each of the four notations being studied).

3.3 Experimental Material & Tasks

In total, three documents were used per representation:

- An *info sheet* about the assigned behavioral constraint representation was made available to the participants one week before the experiment execution for preparation purposes. The descriptions used in these documents are based on the pattern descriptions provided by *Declare* and the *Property Specification Patterns*.[6] To keep the number of language elements to remember approachable, the experiment design considers limitations in human capacity for processing information (cf. Miller [53]). That is, the info sheets of all groups were limited to introducing at most nine language elements. The experiment itself was similar to a closed book exam, so no additional means of help were allowed. This step was taken to ensure unbiased testing of the participants' understanding of the textual terms and graphical shapes of a notation under the exclusion of potential effects resulting from looking up graphical shapes or textual terms.
- A *question sheet* consisting of general questions on the background of the participant (age, gender, level of education, years of work experience, etc.), the experimental tasks, and a Likert scale-based questionnaire to gain insights on how the different representations are subjectively perceived (e.g., perceived learning difficulty) was handed out at the beginning of the experiment session.
- An *answer sheet* accompanied the question sheet for marking the answers to the questions of the experimental tasks. This document makes an automated evaluation by the e-learning platform *Moodle* possible.[7]

For the creation of the tasks of the experiment, we used an algorithm that randomly generates traces and computes the correct truth value of a constraint (i.e., fitting to the trace) automatically. The implementation makes use of the Event Processing Language (EPL) [27] to encode the behavioral constraint patterns in the Complex Event Processing (CEP) engine *Esper*.[8] Truth values were automatically randomly altered to another truth value to create both wrong and correct answer choices. After that automated generation of the task, we manually checked each answer choice to make sure that correct and incorrect answer choices will be treated in the right way (i.e., wrong answer choices are treated as incorrect and correct answer choices are indeed treated as correct) during the automated processing by Moodle.

In total, there were 18 experimental tasks, each consisting of a behavioral constraint, and the instruction to select the correct answers in the answer sheet and to keep time records. Per task six multiple choice answer options were available, each of them consisting of an execution trace and a (correct or incorrect) truth value. For each option the participant had to decide whether it is correct or incorrect (i.e., whether the truth value is correct for the given trace). Figure 6 shows the first task for each of the four groups, which is based on the Succession pattern. Please note that the instruction text and the table for time tracking is only shown in Figure 6 (a) and omitted in (b), (c), and (d). In case a participant works on a task several times, the time tracking table offers not just

[6] http://www.win.tue.nl/declare/, http://patterns.projects.cs.ksu.edu/
[7] http://moodle.org
[8] http://www.espertech.com/esper/

Table 1. Realization of *Declare* patterns by combining available PSP patterns

Declare Constraint	**PSP Constraint**
choice(*A*, *B*)	*A* occurs or *B* occurs
exclusive_choice(*A*, *B*)	(*A* occurs and *B* never occurs) or (*B* occurs and *A* never occurs)
responded_existence(*A*, *B*)	before *A* [*B* occurs] or after *A* [*B* occurs]
co-existence(*A*, *B*)	(*A* occurs and *B* occurs) or (*A* never occurs and *B* never occurs)
not_succession(*A*, *B*)	before *B* [*A* never occurs] and after *A* [*B* never occurs]
not_co-existence(*A*, *B*)	after *A* [*B* never occurs] and after *B* [*A* never occurs]

a single column, but four columns with four separate start and end times. Instead of letters that may be suggestive of a chronological order of events by the alphabetical order (after "A" comes "B") of the used letters, we use the abstract concepts "space" and "time" (cf. behavioral constraints in Figure 6), which do not indicate any kind of chronological order. In Figure 6 (a), the answer choices c) and d) are correct.

In case of monitoring a behavioral constraint in a system at runtime, it might be the case that it is not only of interest if a specification is satisfied or violated but also whether further state changes are possible that could resolve or cause a violation of a specification. That is, the state of a specification can be either temporary (i.e., the state may change) or permanent (i.e., the state may not longer change). Consequently, to enable a more fine-grained analysis of the participants' understanding of behavioral constraints in the experiment, we employ the concept of runtime states (cf. Bauer et al. [4, 5]) which support four truth value states. In particular, a behavioral constraint at runtime is either `temporarily satisfied`, `temporarily violated`, `permanently satisfied`, or `permanently violated`. Several existing studies make use of the concept of four LTL truth value states (cf. Pešić et al. [60], De Giacomo et al. [18], Maggi et al. [51], Falcone et al.[28], Joshi et al. [43], Morse et al. [56], to name but a few).

To reduce chances of misbehavior, the order of the answer choices was randomized between the experimental groups (cf. Figure 6 (a)-(d)). That is, the answer choices remained the same in each group, only their order of presentation was different. Moreover, in the design of the experiment, orientation variations (i.e., the connector shapes were also presented rotated 180 degrees) in the pattern presentation (cf. Figure 7 and Figure 6 (b)) were considered since the orientation possibly has an impact on understandability. However, with regards to orientation variations, the results did not reveal any conclusive impact on understandability.

Since the Succession pattern is not explicitly covered in PSP, it was realized by a combination of the Response and Precedence patterns (cf. Figure 6 (a)). Table 1 summarizes other *Declare* patterns that are represented in PSP by combining available PSP patterns.

To support a replication of the study, we made the experimental material available online (cf. Czepa & Zdun [15]).

3.4 Hypotheses, Parameters, and Variables

Primarily, this controlled experiment focuses on the following hypotheses:

- $H_{0,1}$: There is no difference in terms of *understandability* between the representations.
- $H_{A,1}$: The approaches differ in terms of their *understandability*.

1) Please keep time records and select the correct answer(s) for the following constraint description:

Start Times (hh:mm:ss)					
End Times (hh:mm:ss)					Task Duration:
Durations (mm:ss)					

(space leads to time) and (space precedes time)

a) At the end of trace ***[space, time, space, space, other]*** the truth value is ***permanently violated***.
b) At the end of trace ***[space, time, time, other, other]*** the truth value is ***permanently satisfied***.
c) At the end of trace ***[space, other, space, other, time]*** the truth value is ***temporarily satisfied***.
d) At the end of trace ***[time, space, time, space, space]*** the truth value is ***permanently violated***.
e) At the end of trace ***[time, space, other, other, other]*** the truth value is ***temporarily satisfied***.
f) At the end of trace ***[other, space, time, space, other]*** the truth value is ***temporarily satisfied***.

(a) PSP group (with instructions and time record table)

a) At the end of trace ***[space, other, space, other, time]*** the truth value is ***temporarily satisfied***.
b) At the end of trace ***[time, space, other, other, other]*** the truth value is ***temporarily satisfied***.
c) At the end of trace ***[space, time, time, other, other]*** the truth value is ***permanently satisfied***.
d) At the end of trace ***[time, space, time, space, space]*** the truth value is ***permanently violated***.
e) At the end of trace ***[other, space, time, space, other]*** the truth value is ***temporarily satisfied***.
f) At the end of trace ***[space, time, space, space, other]*** the truth value is ***permanently violated***.

(b) DG group (instructions and time record table omitted)

succession(space, time)

a) At the end of trace ***[space, time, space, space, other]*** the truth value is ***permanently violated***.
b) At the end of trace ***[time, space, time, space, space]*** the truth value is ***permanently violated***.
c) At the end of trace ***[space, other, space, other, time]*** the truth value is ***temporarily satisfied***.
d) At the end of trace ***[space, time, time, other, other]*** the truth value is ***permanently satisfied***.
e) At the end of trace ***[time, space, other, other, other]*** the truth value is ***temporarily satisfied***.
f) At the end of trace ***[other, space, time, space, other]*** the truth value is ***temporarily satisfied***.

(c) DT group (instructions and time record table omitted)

a) At the end of trace ***[space, time, time, other, other]*** the truth value is ***permanently satisfied***.
b) At the end of trace ***[time, space, other, other, other]*** the truth value is ***temporarily satisfied***.
c) At the end of trace ***[space, time, space, space, other]*** the truth value is ***permanently violated***.
d) At the end of trace ***[other, space, time, space, other]*** the truth value is ***temporarily satisfied***.
e) At the end of trace ***[space, other, space, other, time]*** the truth value is ***temporarily satisfied***.
f) At the end of trace ***[time, space, time, space, space]*** the truth value is ***permanently violated***.

(d) DGT group (instructions and time record table omitted)

Fig. 6. Example of a task in all four group variants (more specifically: *Task 1* - based on the *Succession* pattern)

Fig. 7. Orientation variation in the presentation of the *Succession* pattern (i.e., the connector shape is rotated 180 degrees)

The understandability construct consists of two dependent variables, namely:

- the *correctness* achieved in trying to mark the correct answers, and
- the *response time*, which is the time it took to complete the 18 tasks.

These dependent variables are commonly used to measure the construct understandability (cf. Feigenspan et al. [29] and Hoisl et al. [38]). The independent variable (also called factor) focuses on the four behavioral constraint representations.

Secondarily, there are hypotheses that are concerned with the participants' opinion on the tested behavioral constraint representations:

- $H_{0,2}$: There is no difference in terms of *perceived learning difficulty* between the representations.
- $H_{A,2}$: The representations differ in terms of *perceived learning difficulty*.

- $H_{0,3}$: There is no difference in terms of *perceived application difficulty* between the representations.
- $H_{A,3}$: The representations differ in terms of *perceived application difficulty*.

- $H_{0,4}$: There is no difference in terms of *personal interest in using the approach* between the representations.
- $H_{A,4}$: The representations differ in terms of *personal interest in using the approach*.

- $H_{0,5}$: There is no difference in terms of *perceived practical application potential* between the representations.
- $H_{A,5}$: The representations differ in terms of *perceived practical application potential*.

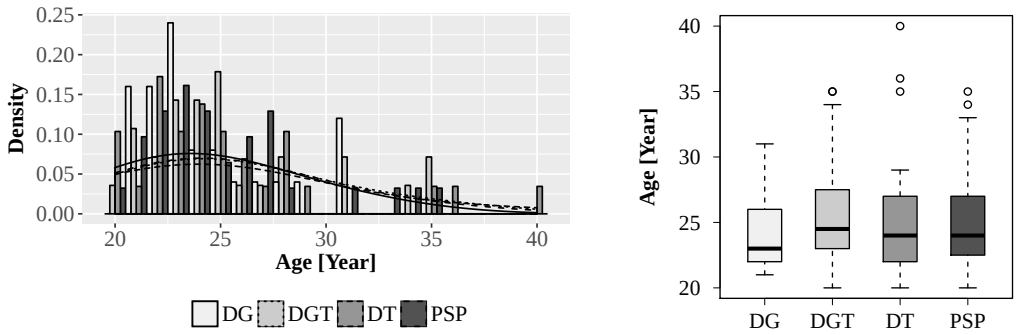
- $H_{0,6}$: There is no difference in terms of *perceived improvement potential* between the representations.
- $H_{A,6}$: The representations differ in terms of *perceived improvement potential*.

3.5 Experiment Design

Wohlin et al. [84] and Kitchenham et al. [46] recommend using a simple experiment design that is appropriate for the goal of a study. In consequence, we applied a completely randomized design with one alternative per participant, which is both a simple design and appropriate for the stated goals (cf. Section 3.1). The participants are assigned to representations in an unbiased manner by using a computerized random allocation to groups.

3.6 Procedure

In total, the experiment had a duration of 90 minutes. The experimental material, namely the question and answer sheet, was provided in the form of printed documents, and the participants were informed about the procedure of the experiment. That involved instructions on how to track time, how to mark answers correctly on the answer sheet, and a pointer to the questionnaire on



(a) Kernel density plot

(b) Box plot

Fig. 8. Participants' age per group

the last page of the question sheet. During the whole experiment session, a clock was displayed by a projector, and the participants were instructed to write down the displayed time when starting and completing work on a task. Seating arrangements were made to limit chances for misbehavior. To limit chances for experimenter bias, the experiment was designed as a multiple-choice test that supports automated processing of the given answers by the e-learning platform *Moodle*. In case of necessary manual interventions (e.g., imprecise markings that we had to clarify), we always made use of the four eyes principle. Moreover, the time recordings and questionnaire answers were processed manually and double-checked subsequently.

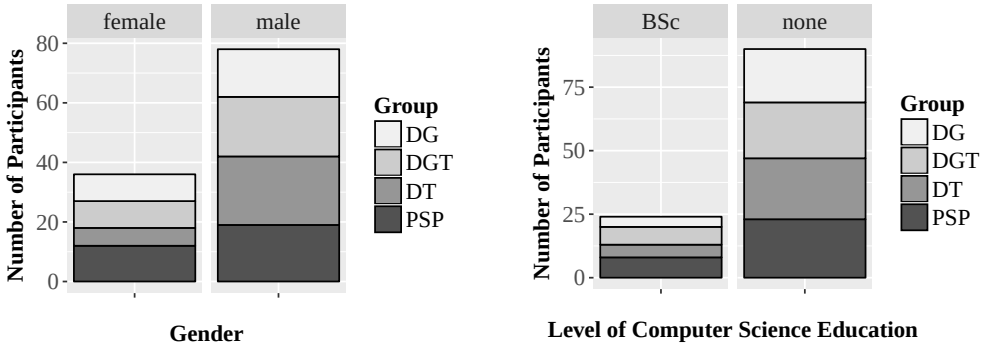
4 ANALYSIS

4.1 Data Set Preparation

The data set was prepared as follows: We had to remove the data of two participants from the data set. One participant used an answer sheet of a different group, which could have been wrongly assigned by the experimenters. To be on the safe side, we decided not to consider this answer sheet as it might have led to confusion (e.g., the results might have accidentally been assigned to the wrong group). The experiment procedure was rigorously implemented in accordance to the planning described in Section 3. Unfortunately, one participant used unauthorized means of aid during the experiment, which has led to the exclusion of this participant from the study. Missing values (5.6% of the dependent variables excluding correctness) were substituted by the arithmetic mean (in case of interval scale data) and median (in case of ordinal scale data) of the data attribute per group.

4.2 Analysis of Previous Knowledge, Experience and Other Features of Participants

In Figure 8, a kernel density plot and box plot of the participants' age per group is shown. The peak density of the participants' age is 23 years, and a high density can be found in the range between 21 and 25 years (cf. Figure 8 (a)). Only very few participants are older than 28 years (cf. Figure 8 (a)), some of them are shown as outliers in the box plot (cf. Figure 8 (b)). The graphical inspection of the data indicates no major differences in the age distribution between the groups. Neither do statistical significance tests indicate any significant differences between the experiment groups (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).



(a) Gender

(b) Level of Education in Computer Science

Fig. 9. Participants' gender and level of computer science education per group

Figure 9 (a) shows a bar chart of the participants' gender distribution. In total, there were 36 female (31.6%) and 78 male participants (68.4%). Inside of the groups, the gender distribution was as follows:

- DG: 9 female (36%) and 16 male participants (64%)
- DGT: 9 female (31%) and 20 male participants (69%)
- DT: 6 female (20.7%) and 23 male participants (79.3%)
- PSP: 12 female (38.7%) and 19 male participants (61.3%)

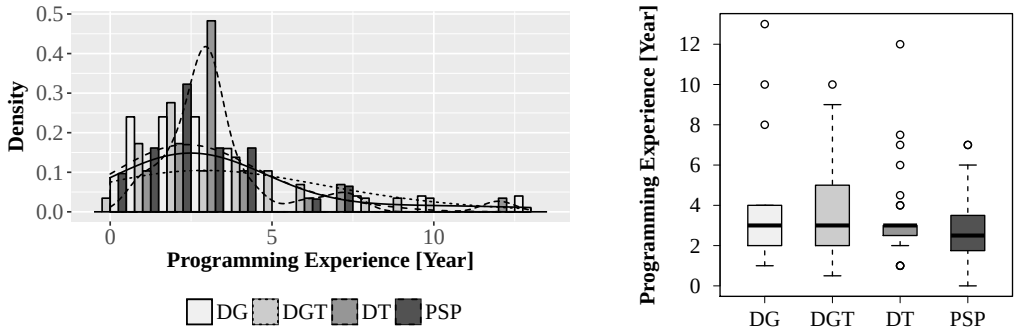
No significant differences were found in gender distribution (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).

The participants' level of education in computer science is shown in Figure 9 (b). Since the courses we recruited the participants from are primarily targeting bachelor students, only 21.1% of the participants hold a Bachelor of Science (BSc) degree in computer science. The distribution between the groups was as follows:

- DG: 4 participants with BSc degree (16%) and 21 participants without any computer science degree (84%)
- DGT: 7 participants with BSc degree (24.1%) and 22 participants without any computer science degree (75.9%)
- DT: 5 participants with BSc degree (17.2%) and 24 participants without any computer science degree (82.8%)
- PSP: 8 participants with BSc degree (25.8%) and 23 participants without any computer science degree (74.2%)

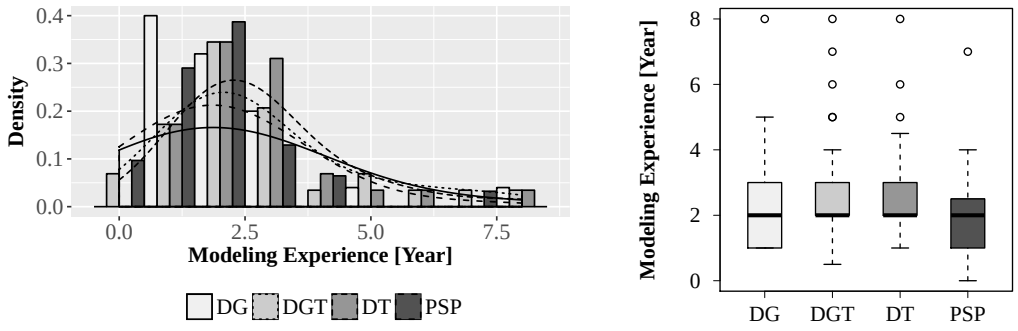
Both the DGT group and PSP group have a slightly larger share of participants with a BSc degree in computer science than the DG and DT groups, but no significant differences were found in level of education between the groups (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).

With regards to programming experience (cf. Figure 10), all groups have a high density in the range of 1 to 4 years of experience. Only very few participants have less than 1 year or more than 4 years of programming experience. Overall, the groups are similar with regards to programming experience. The steeper distribution shape of the DT group appears to be no major difference since we could not find any significant difference in programming experience between the groups (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).



(a) Kernel density plot (b) Box plot

Fig. 10. Participants' programming experience per group



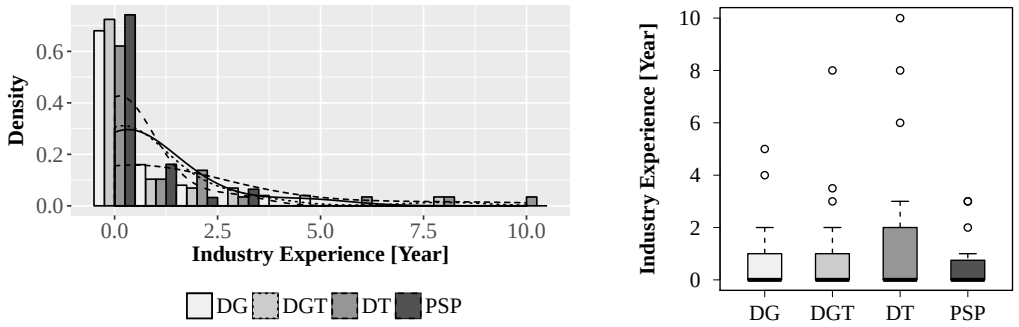
(a) Kernel density plot (b) Box plot

Fig. 11. Participants' modeling experience per group

In the majority of cases, the participants' modeling experience is between 1 and 3 years (cf. Figure 11). There are no significant differences in modeling experience between the groups (all $p > 0.05$ when p-values are adjusted to take multiple testing into account [6]); test applied: two-sided Cliff's delta [11, 70]).

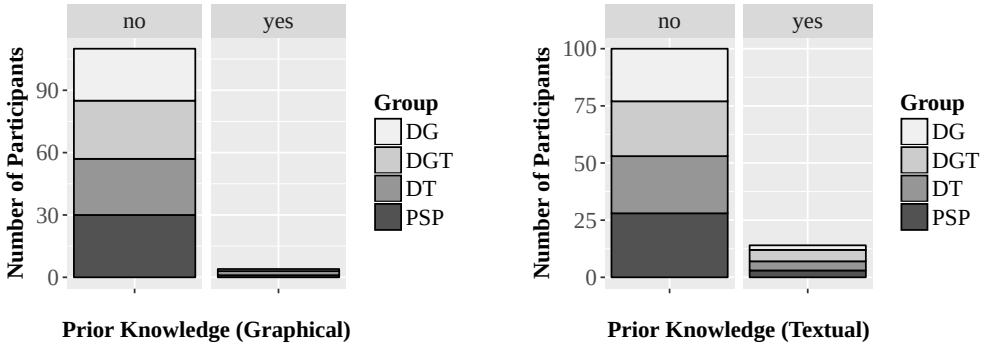
Since the computer science curricula at the University of Vienna are designed for full time studying, the majority of the participants have little to no work experience in the software industry. Some of the students work beside studying or had been working for years in the software industry prior to becoming a computer science student. These circumstances are very well reflected in Figure 12. The industry experience of the different groups appears to be similarly low. There are no significant differences in industry experience between the groups (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).

Almost all participants did not have any prior knowledge of graphical pattern-based behavioral constraint approaches (e.g., Declare [61], Compliance Rule Graphs [50], or Dynamic Condition Response Graphs [25]), and still a great majority of the participants did not have any prior knowledge of textual pattern-based behavioral constraint approaches (e.g., Property Specification Patterns [24],



(a) Kernel density plot (b) Box plot

Fig. 12. Participants' industry experience per group



(a) Graphical approaches (b) Textual approaches

Fig. 13. Participants' prior knowledge of graphical and textual pattern-based behavioral constraint approaches per group

or the Compliance Request Language [25]), as can be seen in Figure 13. The share of participants with prior knowledge of pattern-based behavioral constraint approaches is as follows:

- DG: textual: 2 participants (8%), graphical: 0 participants (0%)
- DGT: textual: 5 participants (17.2%), graphical: 1 participant (3.4%)
- DT: textual: 4 participants (13.8%), graphical: 2 participants (6.9%)
- PSP: textual: 3 participants (9.7%), graphical: 1 participant (3.2%)

There are no significant differences with regards to prior knowledge of graphical and textual pattern-based behavioral constraint approaches between the groups (all $p > 0.05$; test applied: two-sided Cliff's delta [11, 70]).

Overall, with the exception of minor differences, which are to be expected in a completely randomized group allocation, the groups are well-balanced. We could not find any significant differences. That is, there are no indications of disturbing effects on the dependent variables that might have resulted from unbalanced groups.

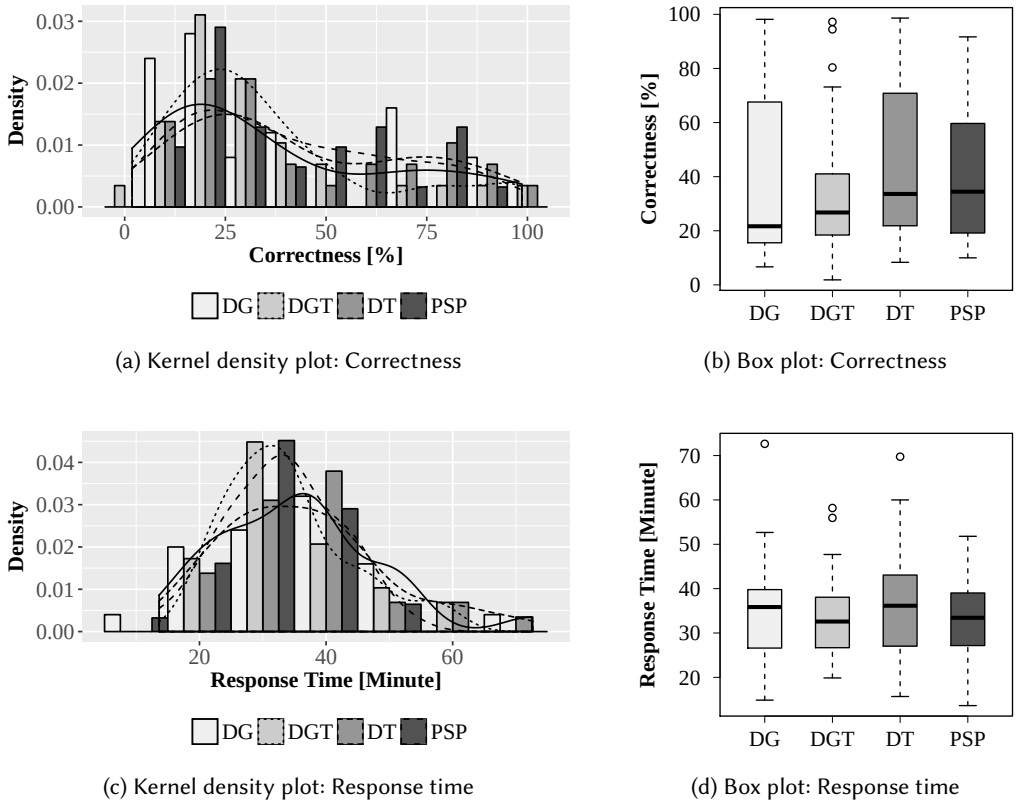
Table 2. Number of observations, central tendency and dispersion of the dependent variables *correctness* and *response time* per group

		DT	DG	DGT	PSP
	Total number of observations	29	26	29	32
	Number of considered observations	29	25	29	31
Correctness	Arithmetic mean [%]	43.64	37.79	34.13	41.46
	Standard deviation (SD) [%]	27.93	29.08	24.86	25.6
	Median [%]	33.61	21.67	26.76	34.44
	Median absolute deviation (MAD) [%]	28.55	17.57	16.89	28.28
	Minimum [%]	8.33	6.67	1.85	10
	Maximum [%]	98.61	98.15	97.22	91.67
	Skew	0.48	0.8	1.17	0.51
	Kurtosis	−1.23	−0.89	0.53	−1.16
Response Time	Arithmetic mean [Minute]	36.28	35.56	33.87	33.09
	Standard deviation (SD) [Minute]	12.88	13.08	9.87	8.85
	Median [Minute]	36.14	35.85	32.58	33.43
	Median absolute deviation (MAD) [Minute]	11.46	11.74	8.75	9.2
	Minimum [Minute]	15.68	14.87	19.87	13.63
	Maximum [Minute]	69.75	72.65	58.18	51.8
	Skew	0.67	0.71	0.79	−0.01
	Kurtosis	−0.03	0.49	−0.08	−0.62

4.3 Descriptive Statistics of Dependent Variables

This section presents the descriptive statistics of the dependent variables. All gathered data have been made publicly available (cf. Czepa & Zdun [15]).

Table 2 contains the number of observations, central tendency and dispersion of the dependent variables *correctness* and *response time* per group. In consequence of the completely random allocation to groups, there were 29 participants in the DT group, 26 participants in DG, 29 participants in DGT, and 32 participants in PSP. Due to irregularities (cf. Section 4.1), we had to exclude the data of one DG participant and one PSP participant. With 43.64 and 41.46 percent, the *correctness arithmetic means* of the DT and PSP groups, which are both purely textual, are between about 4 to 10 percent higher than those of the DG (37.79 percent) and DGT (34.13 percent) groups. Also the *median correctness* values of the PSP (34.44 percent) and DT (33.61 percent) groups are between about 7 to 12 percent higher than those of the DG (21.67 percent) and DGT (26.76 percent) groups. According to the mean and median values, the *response times* appear to be slightly (about 2-3 minutes) faster in the PSP and DGT groups than in the DT and DG groups. Interestingly, many participants achieved a rather low level of correctness while the response times are overall far below the 90 minutes limit in all experiment groups. That is, time was no limiting factor and cannot be the cause of the low achieved correctness scores. The results show large differences in range between the minimum and maximum correctness. We commonly observed such large ranges in course exercises over the past years. Consequently, the results of this study in that regard are aligned with these past observations. Almost all *skew* values are positive, which indicates a right-tailed distribution. With a small negative skew value of −0.01, the PSP response time distribution is rather symmetric. *Kurtosis* is another measure for the shape of a distribution which focuses on the general tailedness. Positive kurtosis values indicate skinny tails with a distribution toward the mean whereas negative kurtosis values indicate fat tails. The majority of the kurtosis values of the correctness variable are negative. The



(a) Kernel density plot: Correctness

(b) Box plot: Correctness

(c) Kernel density plot: Response time

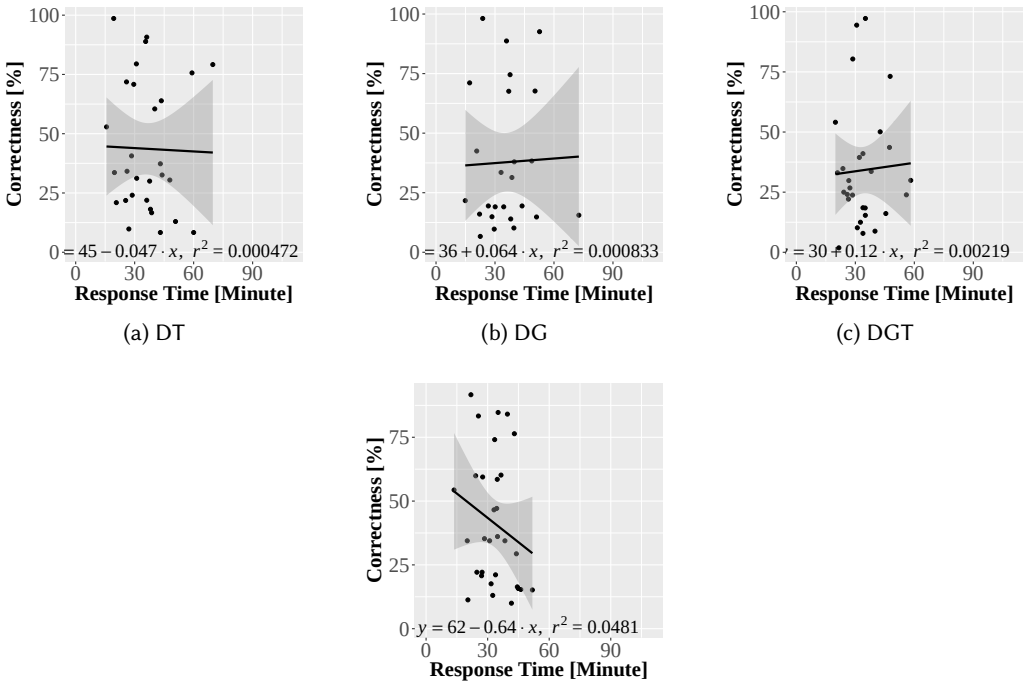
(d) Box plot: Response time

Fig. 14. Kernel density plots and box plots of the dependent variables *correctness* and *response time* per group

sole exception is the DGT kurtosis value of 0.53, which clearly indicates a steeper distribution than in the other groups. With kurtosis values close to zero, the DT (−0.03) and DGT (−0.08) response time distributions are normal-tailed. In contrast, the DG response time distribution has a positive value (0.49) indicating skinny tails, and the PSP response time distribution has a negative value (−0.62) indicating fat tails.

In Figure 14, kernel density plots and box plots of the dependent variables *correctness* and *response time* are shown. The correctness distribution of the DGT group is steeper than those of the other groups (cf. Figure 14 (a)). There are three outliers in the DGT group indicating that only a few participants were able to achieve high levels of correctness in this group (cf. Figure 14 (b)). The outlier at 80.37% correctness had prior knowledge of graphical and textual behavioral constraint approaches while the other two outliers with correctness values of 97.2% and 94.4% did not have any prior knowledge of graphical and textual behavioral constraint approaches. According to the kernel density plot in Figure 14 (c), both the DGT and PSP response time distributions appear to be steeper than those of the other groups. In total there are four response time outliers, all of them in the *Declare* groups (two in DGT and one each in DT and DG).

The scatter plots of the dependent variables *correctness* and *response time* do not show any clear signs of correlation (cf. Figure 15). Moreover, the results of all Kendall's rank correlation tau tests



(a) DT (b) DG (c) DGT

(d) PSP

Fig. 15. Scatter plots of response time vs. correctness with linear trend lines, 95% confidence regions, and coefficients of determination (r^2)

Table 3. Kendall's rank correlation tau of the dependent variables *correctness* and *response time*

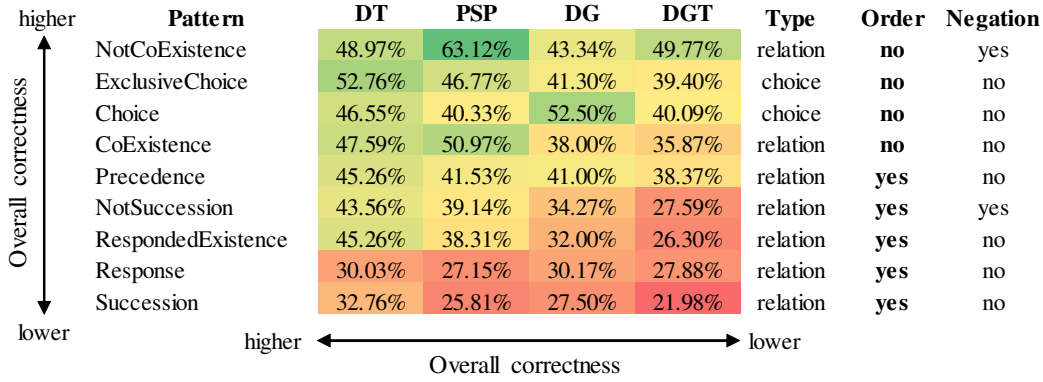
	DT	**DG**	**DGT**	**PSP**
z	−0.8067	0.2804	0.0939	−1.1395
p	0.4198	0.7792	0.9252	0.2545
tau	−0.106	0.0401	0.0124	−0.1447

are non-significant (cf. Table 3). Consequently, there appears to be no correlation between those dependent variables.

For a more detailed analysis of the correctness variable, we make use of a color scale plot (cf. Figure 16) to identify potentially problematic language elements. Interestingly, relations in which the order of the involved states is of importance (e.g., succession) result in lower mean correctness values than patterns in which the order is not of importance (e.g., choice). Especially, the response and succession patterns show a low level of correctness in all groups.

Figure 17 shows diverging stacked bar charts (cf. Heiberger & Robbins [36]) of all Likert responses. In the following, we will discuss them:

- **Statement 1: Studying the behavioral constraint representation approach has been difficult.** 48% of the PSP participants strongly agree or agree that the approach is difficult to learn, followed by DT with 41%, and DGT with 38%. The share of strongly agree answers is



Pattern	DT	PSP	DG	DGT	Type	Order	Negation
NotCoExistence	48.97%	63.12%	43.34%	49.77%	relation	**no**	yes
ExclusiveChoice	52.76%	46.77%	41.30%	39.40%	choice	**no**	no
Choice	46.55%	40.33%	52.50%	40.09%	choice	**no**	no
CoExistence	47.59%	50.97%	38.00%	35.87%	relation	**no**	no
Precedence	45.26%	41.53%	41.00%	38.37%	relation	**yes**	no
NotSuccession	43.56%	39.14%	34.27%	27.59%	relation	**yes**	yes
RespondedExistence	45.26%	38.31%	32.00%	26.30%	relation	**yes**	no
Response	30.03%	27.15%	30.17%	27.88%	relation	**yes**	no
Succession	32.76%	25.81%	27.50%	21.98%	relation	**yes**	no

Fig. 16. Color scale plot of mean correctness per pattern and representation (the greener the more correct, the redder the less correct)

about two times higher in the PSP group than in the DGT and DG groups. The purely graphical DG approach has the lowest percentage of (strongly) agree answers (36%). Interestingly, none of the DT participants answered with strongly agree. According to the bar chart, the purely graphical DG approach appears to be perceived slightly less difficult to learn than the other approaches.

- **Statement 2: Applying the knowledge about the behavioral constraint representation approach has been difficult.** According to the gathered data, it appears to be more difficult to apply the approaches than learning them. With 81%, the share of (strongly) agree answers is higher in the PSP group than in the other groups, followed by the DGT group with 76%. The DG (with 16% disagreeing and 64% (strongly) agreeing) and DT approaches (with 14% (strongly) disagreeing and 69% (strongly) agreeing) are overall perceived a little less difficult to apply than PSP (with 6% disagreeing) and DGT (with 3% disagreeing).
- **Statement 3: I am personally interested in the approach and would like to use it in the (near) future.** With 59%, the majority of DT participants does not show any interest in using the approach in the future. The share of neutral answers is largest in the PSP and DGT groups (52% and 48%), which indicates that the participants of this group are rather undecided. There is, however, a tendency towards (strongly) disagreeing with a share of about 1/3 of the given answers in those groups. Also the DG group shows with a share of 40% (strongly) disagree answers a rather negative attitude towards adopting the approach.
- **Statement 4: The behavioral constraint representation approach can be applied in practice.** With 55% (strongly) agreeing, the PSP group has the largest share of positive answers. Interestingly, the share of strongly agree answers is smaller than in the other three groups, and the PSP group has no strongly disagree answers at all. DG is second with 48% positive and only 8% negative answers, followed by DT with 45% (strongly) agreeing and 8% (strongly) disagreeing. 55% of the DGT participants are undecided, but there is a clear tendency towards (strongly) agreeing (38%).
- **Statement 5: The behavioral constraint representation approach can be further improved.** The share of (strongly) agree answers is large ($\geq$ 60%) in all groups. At the same time, there are no strongly disagree answers and the share of disagree answers is very low (6% in PSP and 3% in DT). The largest share of (strongly) agree answers is present in the DGT group (72%).

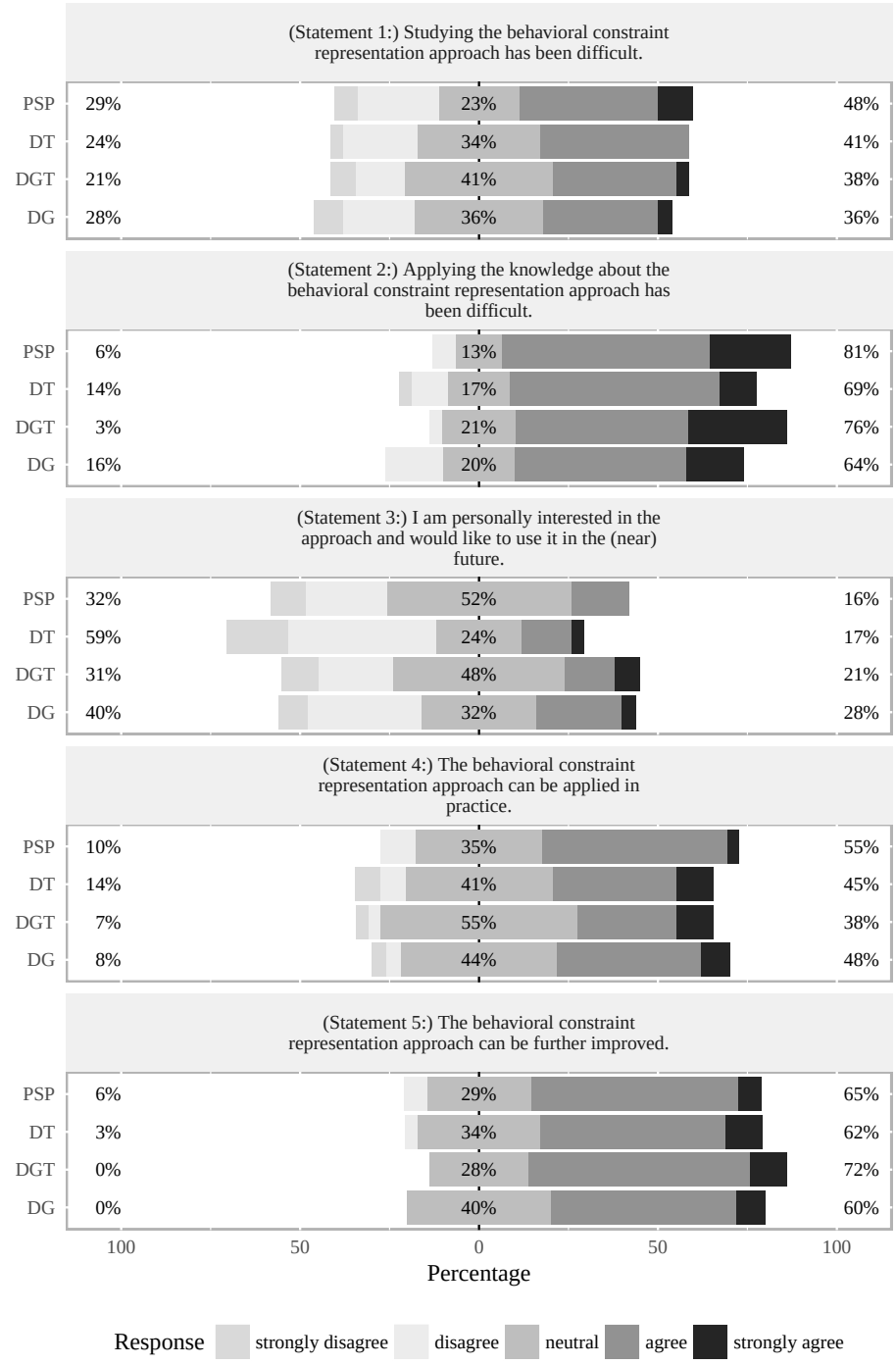
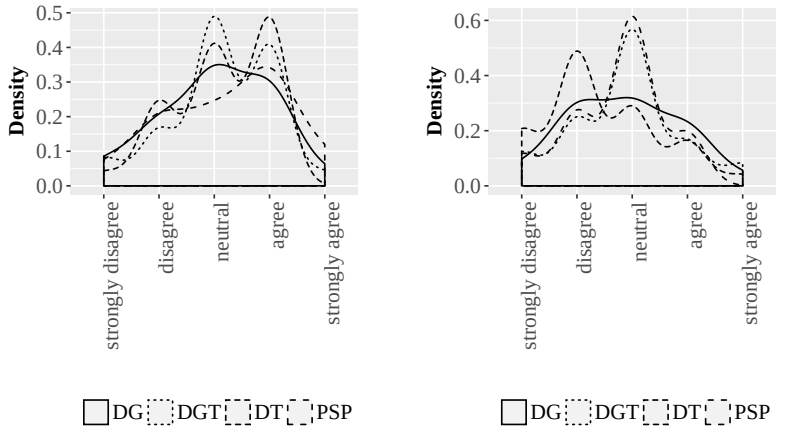


Fig. 17. Diverging stacked bar charts of participants' Likert responses



(a) Statement 1 ("difficult to learn")

(b) Statement 3 ("personally interested in using the approach")

Fig. 18. Selected kernel density plots of Likert responses

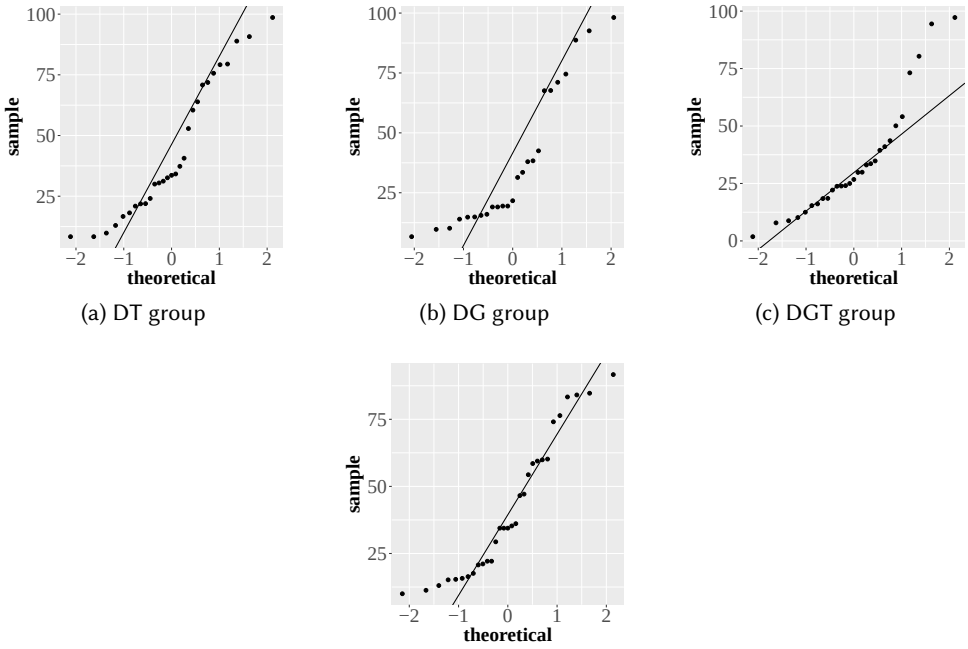
Table 4. Shapiro-Wilk test of normality (* for $\alpha = 0.05$, ** for $\alpha = 0.01$)

	DT	**DG**	**DGT**	**PSP**
Correctness	$W = 0.90205$ $p = 0.008112$ **	$W = 0.84027$ $p = 0.001158$ **	$W = 0.90796$ $p = 0.01528$ *	$W = 0.86744$ $p = 0.001775$ **
Response Time	$W = 0.98841$ $p = 0.9783$	$W = 0.95337$ $p = 0.2982$	$W = 0.95585$ $p = 0.2587$	$W = 0.93301$ $p = 0.06584$

In addition to the visualization by diverging stacked bar charts in Figure 17, we are interested in the shape of the distributions as well, as they are important for testing model assumptions of statistical tests. Figure 18 shows kernel density plots of the Likert responses with especially striking differences in distribution shape. In Figure 18 (a) the PSP and DG distribution shapes are less steep than those of the other two approaches, and the PSP group has its peak at *agree* whereas the the DG group has its peak at *neutral*. In Figure 18 (b), the DG distribution shape is rather flat in comparison to the remaining distribution shapes. The DT group has its peak at *disagree* while the DGT and PSP approaches show a similar distribution shape and location with a peak at *neutral*.

5 STATISTICAL INFERENCE

For proper hypothesis testing, it is important to select the most suitable method. Particularly, it is preferable to choose the method with the greatest statistical power given the properties of the data. Specific model assumptions must be met. A crucial model assumption of parametric testing is normality. The graphical analysis by normal Q-Q plots (cf. Figure 19) and Shapiro-Wilk tests of normality (cf. Table 4) suggest that the normality assumption does not hold in multiple cases. Specifically, the normality assumption does not hold for the correctness variables of all groups. Since there are indications of non-normality in the metric dependent variables *correctness* and *response time* (cf. Section 4.3), the model assumptions for parametric testing are violated. Therefore,



(a) DT group (b) DG group (c) DGT group

(d) PSP group

Fig. 19. Normal QQ plots of correctness data

parametric testing is ruled out. The non-parametric Kruskal-Wallis test assumes that the distribution shapes do not differ apart from their central locations. The relevant descriptive statistics of the data (cf. Section 4.3), namely the skew/kurtosis values and the kernel density plots, suggest differences in the shape of distribution between the groups. Due to the properties of the data, we make use of *Cliff's delta* (cf. Cliff [11] and Rogmann [70]), a robust non-parametric test that is unaffected by change in distribution and non-normal data. Neither of the test results is significant at the $\alpha = 0.05$ level (cf. Table 5 and Table 6). Consequently, the null hypotheses $H_{0,1}$ to $H_{0,6}$ (cf. Section 3.4) cannot be rejected.

The statistics software *R* was used for all statistical analyses.[9] In particular, we used the following libraries in the course of our statistical evaluations: *biotools* [17], *car* [31], *ggplot2* [83], *mvnormtest* [75], *mvoutlier* [63], *orddom* [70], *psych* [68], *usdm* [57], and *likert* [41].

6 ANALYSIS OF FREE TEXT ANSWERS

In addition to the 18 experiment tasks and the Likert scale-based questionnaire, we asked three free text questions to capture the thoughts of the participants regarding the studied and applied behavioral constraint representation. These questions focused on the personal opinion of the participants regarding positive ("likes") and negative ("dislikes") aspects of the assigned behavioral constraint approach as well as suggestions for improvement. In particular, the following three questions were asked:

- What do you like about the behavioral constraint representation approach?

[9] https://www.r-project.org/

Table 5. Cliff's d of the dependent variables *correctness* and *response time*, two-tailed with confidence intervals calculated for $\alpha = 0.05$ (cf. Cliff [11] and Rogmann [70])

		DT vs. **DG**	**DT** vs. **DGT**	**DT** vs. **PSP**	**DG** vs. **DGT**	**DG** vs. **PSP**	**DGT** vs. **PSP**
Correctness	$p_1 = P(X > Y)$	0.5807	0.5898	0.515	0.4966	0.4335	0.4171
	$p_2 = P(X = Y)$	0	0.0024	0.0001	0.0013	0	0.0034
	$p_3 = P(X < Y)$	0.4193	0.4078	0.4849	0.5021	0.5665	0.5795
	d	−0.1614	−0.1819	−0.03	0.0055	0.1329	0.1624
	s_d	0.1599	0.151	0.1521	0.1629	0.16	0.1498
	z	−1.0089	1.2049	−0.1974	0.0339	0.8309	1.0839
	CI low	−0.4516	−0.4557	−0.3188	−0.3057	−0.1851	−0.1386
	CI high	0.1597	0.1233	0.2639	0.3156	0.4257	0.436
	p	0.3177	0.2333	0.8442	0.9731	0.4098	0.2829
Response Time	$p_1 = P(X > Y)$	0.5103	0.5565	0.5551	0.5421	0.5471	0.5006
	$p_2 = P(X = Y)$	0	0	0	0	0	0
	$p_3 = P(X < Y)$	0.4897	0.4435	0.4449	0.4579	0.4529	0.4994
	d	−0.0207	−0.113	−0.1101	−0.0841	−0.0942	−0.0011
	s_d	0.1597	0.1547	0.1521	0.1634	0.1605	0.1513
	z	−0.1296	−0.7299	−0.7242	−0.5149	−0.5867	−0.0073
	CI low	−0.3239	−0.399	−0.3919	−0.3875	−0.3917	−0.291
	CI high	0.2864	0.1932	0.1905	0.2357	0.2211	0.2889
	p	0.8974	0.4685	0.4719	0.6088	0.5598	0.9942

- What do you dislike about the behavioral constraint representation approach?
- How would you improve the behavioral constraint representation approach?

Our analysis of the textual answers of the participants has been inspired by the summative content analysis approach [40]. Since the majority of answers given by the participants is very short and in note form, running a full-blown summative content analysis, which usually focuses on journal manuscripts or specific content in textbooks, is impossible. Nevertheless, it is possible to use the core idea of the technique, namely the counting of occurrences of specific content and the interpretation of the context associated with its use. In the following, we present the results of this analysis:

- 17.5% of all participants (13.8% in DT, 20% in DG, 17.2% in DGT, and 19.4% in PSP) have shown interest in practical examples and case studies to deepen the knowledge and to grasp the full potential of their assigned approach, especially when applied in real-world scenarios.
- 16.7% of all participants (20.7% in DT, 8% in DG, 27.6% in DGT, and 9.7% in PSP) stated that they prefer a more formal definition of the available patterns (and scopes) of the assigned behavioral constraint representation to alleviate ambiguities that are inherently present in natural language.
- Three participants (9.7%) of the PSP group and one participant (3.4%) of the DT group reported issues regarding understanding truth values, especially the difference between temporary and permanent states, whereas one participant (3.4%) of the DGT group mentioned truth values positively and another one negatively. Neither positive nor negative aspects regarding truth values were mentioned by any participant of the DG group.

Table 6. Cliff's d of Likert scale responses, two-tailed with confidence intervals calculated for $\alpha = 0.05$ (cf. Cliff [11] and Rogmann [70])

		DT vs. **DG**	**DT** vs. **DGT**	**DT** vs. **PSP**	**DG** vs. **DGT**	**DG** vs. **PSP**	**DGT** vs. **PSP**
Statement 1	$p_1 = P(X > Y)$	0.3779	0.3424	0.327	0.3255	0.3187	0.3382
	$p_2 = P(X = Y)$	0.3007	0.3163	0.287	0.2938	0.2594	0.2658
	$p_3 = P(X < Y)$	0.3214	0.3413	0.386	0.3807	0.4219	0.396
	d	−0.0566	−0.0012	0.059	0.0552	0.1032	0.0578
	s_d	0.1525	0.1451	0.1438	0.1524	0.1492	0.1451
	z	−0.3709	−0.0082	0.4099	0.3621	0.6919	0.3987
	CI low	−0.3443	−0.2802	−0.2223	−0.2419	−0.1919	−0.2255
	CI high	0.2409	0.278	0.3311	0.3429	0.3812	0.3322
	p	0.7122	0.9935	0.6834	0.7187	0.492	0.6915
Statement 2	$p_1 = P(X > Y)$	0.3255	0.2224	0.2047	0.2386	0.2297	0.3203
	$p_2 = P(X = Y)$	0.349	0.3507	0.3926	0.3228	0.3509	0.3716
	$p_3 = P(X < Y)$	0.3255	0.4269	0.4027	0.4386	0.4194	0.3081
	d	0	0.2045	0.198	0.2	0.1897	−0.0122
	s_d	0.1479	0.1361	0.1302	0.1452	0.1422	0.1386
	z	0	1.5028	1.5205	1.3774	1.3338	−0.0883
	CI low	−0.2846	−0.0732	−0.0673	−0.096	−0.0993	−0.2787
	CI high	0.2846	0.4528	0.4372	0.4635	0.4491	0.2559
	p	1	0.1385	0.1338	0.1743	0.1878	0.9299
Statement 3	$p_1 = P(X > Y)$	0.2621	0.2592	0.2681	0.36	0.3755	0.3604
	$p_2 = P(X = Y)$	0.2579	0.2414	0.2569	0.2648	0.2839	0.3281
	$p_3 = P(X < Y)$	0.48	0.4994	0.475	0.3752	0.3406	0.3115
	d	0.2179	0.2402	0.2069	0.0152	−0.0348	0.0489
	s_d	0.1486	0.1443	0.1443	0.1543	0.1534	0.1407
	z	1.4665	1.6649	1.4336	0.0983	−0.2271	−0.348
	CI low	−0.0865	−0.057	−0.0873	−0.282	−0.3258	−0.3164
	CI high	0.4851	0.4983	0.4679	0.3097	0.2621	0.2257
	p	0.1485	0.1015	0.1571	0.9221	0.8212	0.7291
Statement 4	$p_1 = P(X > Y)$	0.3103	0.3377	0.2959	0.3517	0.3006	0.2781
	$p_2 = P(X = Y)$	0.3338	0.3389	0.3348	0.3642	0.3691	0.3448
	$p_3 = P(X < Y)$	0.3559	0.3234	0.3693	0.2841	0.3303	0.3771
	d	0.0455	−0.0143	0.0734	−0.0676	0.0297	0.099
	s_d	0.1485	0.1433	0.1419	0.147	0.1459	0.141
	z	0.3065	0.0996	0.5174	−0.4597	0.2034	0.702
	CI low	−0.2438	−0.2891	−0.2053	−0.3451	−0.2534	−0.1801
	CI high	0.3274	0.2627	0.3411	0.2208	0.3081	0.3633
	p	0.7605	0.921	0.6069	0.6477	0.8396	0.4855
Statement 5	$p_1 = P(X > Y)$	0.3021	0.2354	0.3026	0.2152	0.2852	0.3348
	$p_2 = P(X = Y)$	0.4151	0.4269	0.4093	0.4414	0.4232	0.4472
	$p_3 = P(X < Y)$	0.2828	0.3377	0.2881	0.3434	0.2916	0.218
	d	−0.0193	0.1023	−0.0145	0.1283	0.0065	−0.1168
	s_d	0.1437	0.1358	0.1359	0.1403	0.1405	0.1302
	z	−0.1344	0.7529	−0.1063	0.9144	0.0459	−0.8969
	CI low	−0.2949	−0.1674	−0.276	−0.1523	0.2651	−0.3615
	CI high	0.2592	0.3577	0.2491	0.3897	0.2771	0.143
	p	0.8936	0.4547	0.9156	0.3647	0.9635	0.3735

- Two participants (6.9%) of the DGT group and one participant (3.4%) of the DT group stated that they would have wanted access to the learning material during the experiment, because they had problems memorizing the meaning of the behavioral constraint patterns.
- 40% of the DG group participants reported problems regarding the graphical notation while 8% mentioned positive aspects. Two DG participants mentioned that the exclusive choice and choice symbols are hard to differentiate. Another participant stated that the `not_succession` pattern was difficult to grasp and remember. One participant reported that it was difficult to remember the order associated with relation patterns. Making use of more symbols rather than using combinations of symbols was proposed by one participant. The feedback of the remaining participants was more general in nature (e.g., "syntax is confusing" or "meaning of signs is hard to understand"). Merely a single participant stated that the shapes used in the graphical representation are clear and easy-to-read. Another participant mentioned his personal preference towards graphical approaches in general.
- The share of DGT participants reporting problems with their assigned representation is 24.1%. Similar to the feedback of one DG participant, one participant would prefer using more shapes for a better differentiation of the patterns. Another participant found the naming of the patterns unclear. In this regard, another participant suggested to use terms present in boolean algebra (e.g., "or" instead of "choice"). The rest of the comments are more general in nature (e.g., "not intuitive"). Two participants (6.9%) mentioned the graphical operators positively ("I liked the graphic representation, as the graphics contained some semantic information about the constraint" and "easy to understand representation in form of simple symbols").
- 20.7% of the DT participants mentioned negative aspects about their assigned textual representation while 10.3% mentioned positive aspects. Like one of the DGT participants, one DT participant desires clearer naming of the patterns. Another participant mentioned that the naming of the patterns is appropriate. Interestingly, one participant stated problems regarding understanding the meaning of the direction of statements (e.g., whether `succession(`A`, `B`)` means A succeeds B or B succeeds A). Originally, we had assumed that the direction would be understood by the reading direction implicitly. However, just a single participant reported this issue, so it is highly questionable whether this is a general issue. Two participant reported difficulties regarding using the approach due to "a lot of similarities between the constraints", which makes them hard to distinguish. Two participants suggested adding a negation operator to the constraint language to support negations for each of the available patterns. Another participant liked that there exists a specific pattern for "every case", but at the same time criticized that the number of constraints is growing rapidly if the implementation of new scenarios becomes necessary. One participant liked the function-like style of the approach, which is familiar to programmers.
- 19.4% of the PSP participants criticized some aspects of their assigned representation while 9.8% mentioned positive aspects. One participant would prefer a "more sophisticated visualization" instead of the textual representation. Another participant was fond of the natural language approach, but criticized the lack of syntax highlighting in the experiment. We wanted to present all four tested representations by similar means to avoid bias towards a specific representation, so syntax highlighting was omitted in the PSP task descriptions intentionally. In actual implementations of the PSP approach, syntax highlighting or similar techniques are usually supported (e.g., Czepa et al. [13]). One participant liked the use of boolean connectors, but would have wanted to see the actual symbols instead of words (e.g., $\wedge$ for `and`). A similar comment was made by another participant who suggests "writing in a more mathematical way". Another participant would have wanted to see the `xor` operator

in use. The difference between the `between` and `after-until` scopes was mentioned as "difficult to understand" by one participant. Another participant mentioned that the scopes and patterns are clearly understandable.

7 DISCUSSION

7.1 Evaluation of Results and Implications

While the descriptive statistics and the results of the analysis of free text answers appear to be slightly in favor of the textual approaches, the results of the inference statistics do not indicate any significant difference between the tested representations. That is, the following null hypotheses cannot be rejected:

- $H_{0,1}$: There is no difference in terms of *understandability* between the representations.
- $H_{0,2}$: There is no difference in terms of *perceived learning difficulty* between the representations.
- $H_{0,3}$: There is no difference in terms of *perceived application difficulty* between the representations.
- $H_{0,4}$: There is no difference in terms of *personal interest in using the approach* between the representations.
- $H_{0,5}$: There is no difference in terms of *perceived practical application potential* between the representations.
- $H_{0,6}$: There is no difference in terms of *perceived improvement potential* between the representations.

However, it is striking that the achieved correctness is rather low on the average. From a prior experiment on the understandability of textual behavioral constraint approaches (cf. Czepa & Zdun [16]), it is evident that higher correctness values (about 70% on the average in PSP) are achievable if access to learning material and other material (e.g., hand-written notes) is granted during the experiment session. That is, it appears to be difficult to deduce the meaning of pattern-based behavioral constraints from their textual and/or graphical representations without additional support. As additional support was given for none of the groups, we do not think that this aspect could have influenced the relative outcomes of the experiment. In this regard, it might be possible that both approaches could benefit from a greater degree of additional support in a similar fashion. The analysis of the given free text answers regarding positive/negative aspects and suggestions for improvement with regard to the achievable correctness levels (cf. Section 6) provides additional evidence. Consequently, there are two angles for improvement, namely the representation itself (i.e., finding better graphical and/or textual representations) and the support provided (i.e., supportive means provided by a behavioral constraint modeling tool).

7.2 Threats to Validity

All known threats that might have an impact on the validity of the results are discussed in this subsection.

7.2.1 Threats to Internal Validity. Threats to internal validity can be described as unobserved variables that might have an unwanted influence on an experiment's result by disturbing the causal relationship of independent and dependent variables. There exist several threats to the internal validity of this experiment, which must be discussed:

- *History effects* refer to events that occur in the environment and change the conditions of a study. The short duration of the study limits the possibility of changes in environmental conditions, and we did not observe any such, but we cannot entirely rule out any such effect,

prior to the study taking place. However, in such a case, it would be extremely unlikely that the scores of one group are more affected than another, because of the random allocation of participants to groups.

- *Maturation effects* refer to the impact that time has on an individual. Since the duration of the experiment was short, maturation effects are considered to be of minor importance, and we did not observe any such effects.
- *Testing effects* comprise learning effects and experimental fatigue. *Learning effects* were avoided by testing each person only once. *Experimental fatigue* is concerned with occurrences during the experiment that exhaust the participant either physically or mentally. The participants did not report, and we did not observe, any signs of fatigue.
- *Instrumental bias* occurs if the measuring instrument (i.e., a physical measuring device or the actions/assessment of the researcher) changes over time during the experiment. We tried to avoid instrumental bias by using an experimental design that enables an automated and standardized processing of the test results.
- *Selection bias* is present if the experimental groups are unequal before the start of the experiment (e.g., severe differences in relevant experience, age, or gender). Usually, selection bias is likely to be more threatening in quasi-experimental research. By using an experimental research design with the fundamental requirement to randomly assignment participants to the different groups of the experiment, we can avoid selection bias to a large extent. Moreover, our evaluation of the composition of the groups (regarding age, gender, experience/education in different dimensions) did not indicate any major differences.
- *Experimental mortality* is only likely to occur if the experiment lasts for a long time because the chances for dropouts increase (e.g., participants moving to another geographical location). Due to the short time frame of this study, experimental mortality was not an issue at all.
- *Diffusion of groups* occurs if a group of the experiment is contaminated in some way by another experiment group. We tried to mitigate this threat by asking the participants not to disclose or discuss anything related to the experiment before the experiment session. Since the participants share the same social group, and they are interacting outside the research process as well, we cannot entirely rule out a cross-contamination between the groups.
- *Compensatory rivalry* is present if participants of a group put in extra effort when they have the impression that the representation of another group might lead to better results than their own. This threat is mitigated by insisting on nondisclosure.
- *Demoralization* could occur if a participant is assigned to a specific group that she/he does not want to be part of. We did not observe any signs of demoralization such as increased dropout rates or complaints regarding group allocation.
- *Experimenter bias* refers to undesired effects on the dependent variables that are unintentionally introduced by the researcher. The experiment was designed in a way that limits chances for this kind of bias. In particular, all participants received a similar training and worked on the same set of tasks (i.e., only the constraint representation differs). Moreover, the results of the controlled experiment were processed automatically in a standardized procedure.
- *Self-selection bias*: The possibility of self-selection bias appears to be negligible as merely three students participated in alternative activities.
- *Impact of preparation*: We designed the preparation material in such a way to keep the necessary effort involved in learning the patterns at a doable level for each participant. In accordance with Miller's law [53], at most nine language elements were presented in each experiment group. Consequently, the learning effort was minimal, which strongly mitigates the risk of insufficient preparation. Moreover, instead of directly asking the degree of effort spent on preparation, which might lead to insincere answers (i.e., a participant

might expect to be punished for not preparing well), we tried to check indirectly by asking how difficult the studying of the approach was. We assumed that a participant who did not prepare himself/herself will not have a strong opinion on that topic and tick the neutral item or abstain. Subsequently, we removed the data of these participants from the data set and performed hypotheses testing. As in the full data set, no test result was significant. That is, even if participants who possibly did not prepare themselves are not considered, the results still apply.

7.2.2 Threats to External Validity. The external validity of a study focuses on its generalizability. In the following, we discuss potential threats that hinder a generalization. Different types of generalizations must be considered:

- *Generalizations across populations*: By statistical inference, we try to make generalizations from the sample to the immediate population. We do not intend to claim generalizability across populations without further empirical evidence. This study has a strong focus on the understandability of the tested representations from the viewpoint of novice software designers. We acknowledge that expert users who are familiar with *Declare* and/or *Property Specification Patterns* potentially perform better.
- *Generalizations across groups*: Since the experiment groups focus on specific behavioral constraint representations, options for variation are limited. Nevertheless, in future studies, it might be interesting to introduce new or amended representations (e.g., a graphical representation that is based on DGT, but using just a single relation shape).
- *Generalizations across settings/contexts*: The participants of this study are students who enrolled in computer science courses at the University of Vienna, Austria. Apparently, a majority of the students are Austrian citizens, but there is a large presence of foreign students as well. Surely, it would be interesting to repeat the experiment in different settings/contexts to evaluate the generalizability in that regard. For example, repeating the experiment with English native speakers might lead to different (presumably better) results since English terms are used in the textual/hybrid constraint representations.
- *Generalizations across time*: In general, it is hard to predict whether the results of this study hold over time. For example, if teaching of graphical or textual behavioral constraint approaches is intensified in the computer science curricula at the University of Vienna, then the students would bring in more expertise, which likely would have an impact on the results.

7.2.3 Threats to Construct Validity. There are potential threats to the validity of the construct that must be discussed:

- *Inexact definition & Construct confounding*: This study has a primary focus on the construct *understandability,* which is measured by the dependent variables *correctness* and *response time*. This construct is exact and adequate. Several existing studies use this construct and its variables (cf. Feigenspan et al. [29] and Hoisl et al. [38]).
- *Mono-method bias*: To measure the correctness of answers, the evaluation by an automated method appears to be the most accurate measure as it does not suffer from experimenter bias or instrumental bias. Keeping time records was the personal responsibility of each participant due to organizational reasons. The participants were instructed extensively on how to keep time records, and they were informed that accurate time record keeping will have a positive impact on the final grading. We also made clear that the overall response time has no influence on the grading to avoid time stress. We did not detect any irregularities (e.g., overlapping time frames or long pauses) in those records. Nonetheless, this measuring method leaves room for measuring errors, and an additional or alternative measuring method (e.g., performing

the experiment with an online tool that handles record keeping) would reduce this threat. The additional task of keeping accurate time records might have had a negative impact on performing the actual experiment tasks, but no participant reported any such effect.
- *Reducing levels of measurements*: Both the correctness and response time are continuous variables. That is, the levels of measurements are not reduced. The Likert scales (also called agree-disagree rating scales) used in this study offer 5 answer categories rather than 7 or 11, because the latter produce data of lower quality according to Revilla et al. [69].
- *Group-sensitive factorial structure*: In some empirical studies a specific assigned experiment group might sensitize participants to develop a different view on a construct. Since we did not ask questions regarding the subjective level of understandability, but tried to measure the actual level of understandability objectively, this threat appears to not be present at all. The questionnaire at the end of the question sheet is neither meant nor used to measure the understandability construct, but used to measure other aspects. Here, we tried to mitigate this threat by focusing on one-dimensional constructs (i.e., the multi-dimensional construct *perceived difficulty* is split up into *perceived learning difficulty* and *perceived application difficulty*).

7.2.4 Threats to Content Validity. Content validity is concerned with the relevance and representativeness of the elements of a study for the construct that is measured:

- *Relevance*: All tasks of this study are based on recurring behavioral constraint patterns that are present in existing graphical and textual behavioral constraint approaches (cf. [24, 25, 61, 80]).
- *Representativeness*: A representative subset of existing behavioral constraint patterns was used for designing the tasks of the experiment. In this study we focused on a set of commonly used binary relations (cf. [24, 25, 61, 80]). Unary constraints are common as well, but we decided to omit them due to their simplicity. It is also worth noting that some behavioral constraints in *Declare* are not covered in the Property Specification Patterns, and vice versa. In particular, the scopes of PSP are not part of Declare, the chain patterns have a different meaning in *Declare* and PSP, and *Declare* supports additional behavioral constraints (i.e., alternate, negative relation and choice patterns). Nevertheless, some of the *Declare* patterns which are not explicitly covered by PSP can be realized by combinations of Property Specification Patterns. Others, like the alternate patterns of *Declare* are not covered by the originally proposed PSP collection. Naturally, it would have been possible for us to extend both *Declare* and PSP with new patterns including new graphical and textual elements, but proposing new pattern representations was not the goal of this empirical study, so the established as-is state of these approaches was covered.

7.2.5 Threats to Conclusion Validity. Retaining outliers might be a threat to conclusion validity. However, all outliers appear to be valid measurements, so deleting them would pose a threat to conclusion validity as well. We performed a thorough investigation of model assumptions before applying the most suitable statistical test with the greatest statistical power, given the properties of the acquired data. That course of action is considered to be extremely beneficial to the conclusion validity of this study.

8 RELATED WORK

We are not aware of any existing empirical studies that investigate the differences in understandability of representative graphical and textual behavioral constraint languages in a similar way and depth as the presented study does.

We also would like to mention that there exists a huge body of studies on understandability of models that are merely remotely related to our work. For example, a study by Reijers and Mendling [67] investigates the understandability of classical flow-driven business process models. Interestingly, professionals could not be distinguished from the students of two participating universities, and the students of one university performed even better than professionals. However, since that study considers flow-driven business processes only, the results are hardly transferable to behavioral constraints that are declarative in nature.

In the following, we focus on studies that are highly related to the presented work, namely studies that are concerned with the understandability of behavioral constraints.

8.1 Empirical Studies on the Understandability of Behavioral Constraint Representations in Software Architecture and Software Engineering

There exist only very few studies on analyzing and comparing different behavioral constraint languages in the field of software architecture and engineering.

An eye-tracking experiment with 28 participants by Sharafi et al. [74] on the understandability of graphical and textual software requirement models did not reveal any statistically significant difference in terms of correctness of the approaches, which is in line with the results of our study. However, software requirement models and behavioral constraints are merely distantly related. The study also reports that the response times of participants working with the graphical representations were slower. Interestingly though, the participants preferred the graphical notation.

Czepa et al. [14] compared the understandability of three languages for behavioral software architecture compliance checking, namely the Natural Language Constraint language (NLC), the Cause-Effect Constraint language (CEC), and the Temporal Logic Pattern-based Constraint language (TLC), in a controlled experiment with 190 participants. The NLC language is simply using the English language for software architecture descriptions. CEC is a high-level structured architectural description language that abstracts the Event Processing Language [27] and enables nesting of cause parts, that observe an event stream for a specific event pattern, and effect parts, that can contain further cause-effect structures and truth value change commands. TLC is a high-level structured architectural description language that abstracts behavioral patterns. Interestingly, the statistical inference of this study suggests that there is no difference in understandability of the tested languages. This could indicate that the high-level abstractions employed bring those structured languages closer to the understandability of unstructured natural language architecture descriptions. Moreover, it might also suggest that natural language leaves more room for ambiguity, which is detrimental for its understanding. Potential limitations of that study are that its tasks are based on common architectural patterns/styles (i.e., a participant possibly recognizes the meaning of a constraint more easily by having knowledge of the related architectural pattern) and the rather small set of involved behavioral constraint patterns (i.e., only very few behavioral constraint patterns were necessary to represent the architecture descriptions).

Hoisl et al. [38] conducted a controlled experiment on three notations for scenario based model tests with 20 participants. In particular, they evaluated the understandability of a semi-structured natural language scenario notation, a diagrammatic scenario notation, and a fully-structured textual scenario notation. According to the authors, the purely textual semi-structured natural language scenario notation is recommended for scenario-based model tests, because the participants of this group were able to solve the given tasks faster and more correctly. That is, the study might indicate that a textual approach outperforms a graphical one for scenario based model test, an effect that our study did not discover for behavioral constraints. However, the validity of the experiment is limited by its sample size and the lack of statistical hypothesis testing.

A controlled experiment carried out by Heijstek et al. [37] with 47 participants focused on finding differences in understanding of textual and graphical software architecture descriptions. Interestingly, participants who predominantly used textual architecture descriptions performed significantly better, which suggests that textual architectural descriptions could be superior to their graphical counterparts. In our study, which has a focus specifically on textual and graphical behavioral constraints instead of software architecture descriptions, such an effect was not measurable.

8.2 Empirical Studies on the Understandability of Behavioral Constraint Representations in Business Process Management

In the field of business process management, there exist studies that evaluate the understandability of declarative business processes which are composed of a set of behavioral constraint patterns. These studies are highly related to our work since they investigate the understandability of pattern-based behavioral constraints in the context of declarative business processes.

Weber et al. [82] carried out a controlled experiment (with 25 and 16 participants) on the impact of varying the levels of pattern-based behavioral constraints in planning and executing a journey. In particular, one group was exposed to only 2 behavioral constraints while another had to take 12 behavioral constraints into account. Interestingly, their statistical analysis does not show any significant difference in understanding. That might indicate that potential users handle varying constraint numbers well, but there also might not be enough measurable difference between 2 and 12 constraints. It would be interesting to evaluate how users cope with larger numbers of constraints (e.g., 25, 50, 100) as well. Moreover, the small sample sizes are a threat to validity of this study.

Zugal et al. [87] investigate the understandability of hierarchies in declarative business processes in an experiment with 9 participants. The results of their research indicate that hierarchies must be handled with care. While information hiding and improved pattern recognition are considered to be positive aspects of hierarchies since the mental effort for understanding a process model is lowered, the fragmentation of processes by hierarchies might lower overall understandability of the process model. Another important finding of their study is that users appear to approach declarative process models in a sequential manner even if the user is definitely not biased by previous experiences with sequential business process models (e.g., BPMN [59]). They conclude that the abstract nature of declarative process models does not seem to fit the human way of thinking. Moreover, they observed that the participants of their study tried to reduce the number of constraints to consider by putting away sheets that describe irrelevant sub-process or by using the hand to hide parts of the process model that are irrelevant. The validity of this study is strongly limited by the extremely small sample size.

Haisjackl et al. [34] investigate the users' understanding of declarative business process models that are composed of a set of ten behavioral constraint patterns with 9 participants. The evaluation seems to be based on the same experimental data as in the work by Zugal et al. [87]. Like in that work, they point out that users tend to read such models sequentially despite the declarative nature of the approach. The larger a model, the more often are hidden dependencies overlooked, which indicates increasing numbers of constraints lower understanding. Moreover, they report that single constraints are overall well understood, but there seem to be problems with understanding the precedence constraint. As the authors point out, this kind of confusion could be related to the graphical arrow-based representation of the constraints where subtle differences decide on the actual meaning. That is, the arrow could be confused with a sequence flow as present in flow-driven, sequential business processes. As previously stated for the work by Zugal et al. [87], the validity of this study is possibly strongly affected by the small sample size.

Haisjackl & Zugal [33] investigated differences between textual and graphical declarative workflows using the *Declare* notation in an empirical study with 9 participants. The evaluation seems to be based on the same experimental data as in the work by Zugal et al. [87] and Haisjackl et al. [34]. This study is highly related to our work presented in this article. The authors state that the results of their study indicate that the graphical representation are advantageous in terms of perceived understandability, error rate, duration, and mental effort, but this conclusion seems to be based merely on descriptive statistics (i.e., arithmetic means and counting occurrences). The lack of hypothesis testing and the small number of participants are severe threats to the validity of this study.

An approach by De Smedt et al. [21] tries to improve the understandability of declarative business process models by revealing hidden dependencies. They conduced an experiment with 95 students. The result suggests that explicitly showing hidden dependencies enables a better understanding of declarative business process models.

A study by Pichler et al. [64] compares the understandability of imperative and declarative business process modeling notations. This study indicates that imperative process models are significantly more understandable than declarative models, but the authors also state that the participants had more previous experience with imperative process modeling than with declarative process modeling. Moreover, the sample size (28 participants) is rather small, which is a threat to validity of this study.

Mendes Cunha et al. [52] try to improve declarative business process modeling by taking the comments of 4 persons into consideration. The resulting language is based on the same behavioral constraint patterns, but it proposes different graphical notations. Obviously, the small number of participants and the lack of evaluation of the proposed alternative graphical elements are serious threats to validity.

9 CONCLUSION AND FUTURE WORK

9.1 Summary

The results of this controlled experiment study with 116 participants did not reveal any significant difference in understandability nor in any other tested aspects (i.e., perceived learning difficulty, perceived application difficulty, personal interest in using the representation, perceived practical applicability, perceived potential for further improvement of the behavioral constraint representations) between graphical, textual, and hybrid behavioral constraint representations. Merely the descriptive statistics and the results of the analysis of free text answers are slightly in favor of the tested textual behavioral constraint approaches. The achieved correctness is rather low on the average in all experiment groups. A prior experiment on the understandability of textual behavioral constraint approaches (cf. Czepa & Zdun [16]) yielded higher correctness values (about 70% on the average in PSP) when access to learning material and other material (e.g., hand-written notes) is granted during the experiment session. That is, it appears to be difficult to deduce the meaning of pattern-based behavioral constraints from their textual and/or graphical representations without additional support. The analysis of the given free text answers regarding positive/negative aspects and suggestions for improvement (cf. Section 6) provides additional evidence in that regard.

9.2 Impact

Since there appears to be no significant difference in understandability of textual and graphical behavioral constraint approaches, the results of this empirical study might indicate that the tested representations can be used interchangeably. However, a major obstacle in this regard could be the overall low level of achieved correctness, which must be further investigated. In response to the low

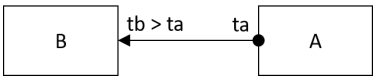


Fig. 20. Alternative response notation stating the temporal order of the involved elements explicitly by labels

level of achieved correctness, this study indicates two angles for further research and improvement of textual and graphical behavioral constraint representations, namely the representation itself (i.e., finding better graphical and/or textual representations) and the technology support provided (i.e., the support provided by a behavioral constraint modeling tool or by analysis, refactoring, and debugging tools).

Our carefully designed and conducted empirical study can work as a solid foundation for further empirical evaluations of pattern-based behavioral constraint representations and their future development.

9.3 Future Work

The experiment could be repeated with different user groups (e.g., industrial practitioners) to gain further insights in the understandability of the representation from different perspectives. Other experiments could be run to further investigate the results. Experiments with different symbols in graphical behavioral constraint representations and variations of the terms used in textual approaches are also opportunities for future research. For example, a new representation could be introduced to the current experimental setup that streamlines the hybrid *Declare* approach (DGT) by reducing the number of available connector shapes to a single shape, just like relations in an ontology. For the evaluation of the understandability of interrelated behavioral constraint collections or the creation process of such, qualitative studies that are based on eye-tracking and think-aloud protocols [26] would further evolve the body of knowledge. Moreover, the presented study focuses on the understandability of already given textual and graphical behavioral constraints, so conducting an experiment on the understandability related to authoring textual and graphical constraints would be another interesting possibility for future research. In this regard, adequate tool support is assumed to be a major topic. The results of this experiment indicate that behavioral constraints in which the order of the involved states is of importance are particularly difficult to understand, so these elements should receive special attention. To improve the understandability of these constraints, a behavioral constraint editor could, for example, provide a tooltip with an animation that illustrates the temporal order of the involved elements (e.g., showing sample traces with the corresponding truth value state). Also, the textual terms and/or graphical elements of the representations should be revisited. For example, the response pattern (both in the textual and/or graphical form) can be easily misunderstood as a strict sequence (e.g., as known from procedural/imperative modeling languages). Such ambiguities must be avoided to achieve higher levels of understanding. An alternative textual representation of the response pattern, which might leave less room for misunderstandings by emphasizing the temporal order, could be *A* `at time` *ta* `requires` *B* `at time` *tb* `>` *ta*. The corresponding amended *Declare* notation of the response pattern is shown in Figure 20. Such amendments can be the starting point for further empirical evaluations with the goal to improve the understandability of behavioral constraint representations.

ACKNOWLEDGMENTS

This work was partially funded by: FFG (Austrian Research Promotion Agency) project CACAO, no. 843461; FWF (Austrian Science Fund) project ADDCompliance: I 2885-N33

REFERENCES

[1] Ahmed Awad, Matthias Weidlich, and Mathias Weske. 2011. Visually specifying compliance rules and explaining their violations for business processes. *Journal of Visual Languages and Computing* 22, 1 (2011), 30 – 55. https://doi.org/10.1016/j.jvlc.2010.11.002 Special Issue on Visual Languages and Logic.

[2] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series).* The MIT Press.

[3] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 1994. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*. Wiley.

[4] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2010. Comparing LTL Semantics for Runtime Verification. *J. Log. and Comput.* 20, 3 (June 2010), 651–674. https://doi.org/10.1093/logcom/exn075

[5] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime Verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.* 20, 4, Article 14 (Sept. 2011), 64 pages. https://doi.org/10.1145/2000799.2000800

[6] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)* 57, 1 (1995), 289–300.

[7] D. Bianculli, C. Ghezzi, C. Pautasso, and P. Senti. 2012. Specification patterns from research to industry: A case study in service-based applications. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. 968–976. https://doi.org/10.1109/ICSE.2012.6227125

[8] B. H. C. Cheng and J. M. Atlee. 2007. Research Directions in Requirements Engineering. In *Future of Software Engineering (FOSE '07)*. 285–303. https://doi.org/10.1109/FOSE.2007.17

[9] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. 2002. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV '02)*. Springer-Verlag, London, UK, UK, 359–364. http://dl.acm.org/citation.cfm?id=647771.734431

[10] Edmund M. Clarke and E. Allen Emerson. 1982. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. In *Logics of Programs*, Dexter Kozen (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 52–71.

[11] N. Cliff. 1993. Dominance statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychological Bulletin* 114 (1993), 494–509.

[12] J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Pasareanu, Robby, and Hongjun Zheng. 2000. Bandera: Extracting Finite-State Models from Java Source Code. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE '00)*. 439–448. https://doi.org/10.1145/337180.337625

[13] Christoph Czepa, Huy Tran, Uwe Zdun, Thanh Tran, Erhard Weiss, and Christoph Ruhsam. 2016. Ontology-Based Behavioral Constraint Authoring. In *2nd International Workshop on Compliance, Evolution and Security in intra- and Cross-Organizational Processes (CeSCoP '16), 20th IEEE International Enterprise Computing Workshops (EDOCW '16)*. http://eprints.cs.univie.ac.at/4754/

[14] Christoph Czepa, Huy Tran, Uwe Zdun, Thanh Tran, Erhard Weiss, and Christoph Ruhsam. 2017. On the Understandability of Semantic Constraints for Behavioral Software Architecture Compliance: A Controlled Experiment. In *IEEE International Conference on Software Architecture (ICSA '17)*. http://eprints.cs.univie.ac.at/5059/

[15] Christoph Czepa and Uwe Zdun. 2018. On the Understandability of Graphical and Textual Pattern-Based Behavioral Constraint Representations [Data set]. https://doi.org/10.5281/zenodo.1209839

[16] Christoph Czepa and Uwe Zdun. 2018. On the Understandability of Temporal Properties Formalized in Linear Temporal Logic, Property Specification Patterns and Event Processing Language. *IEEE Transactions on Software Engineering* (2018), 1–1. https://doi.org/10.1109/TSE.2018.2859926

[17] A. R. da Silva, G. Malafaia, and I. P. P. de Menezes. 2017. Biotools: An R Function to Predict Spatial Gene Diversity via an individual-based approach. *Genetics and Molecular Research* 16 (2017).

[18] Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio Maria Maggi, and Marco Montali. 2014. Monitoring Business Metaconstraints Based on LTL and LDL for Finite Traces. In *Business Process Management*, Shazia Sadiq, Pnina Soffer, and Hagen Völzer (Eds.). Springer International Publishing, Cham, 1–17.

[19] Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI '14)*. AAAI Press, 1027–1033. http://dl.acm.org/citation.cfm?id=2893873.2894033

[20] Giuseppe De Giacomo and Moshe Y. Vardi. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*. AAAI Press, 854–860. http://dl.acm.org/citation.cfm?id=2540128.2540252

[21] Johannes De Smedt, Jochen De Weerdt, Estefanía Serral, and Jan Vanthienen. 2016. Improving Understandability of Declarative Process Models by Revealing Hidden Dependencies. In *Advanced Information Systems Engineering: 28th International Conference, CAiSE '16, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, Selmin Nurcan, Pnina Soffer, Marko Bajec, and Johann Eder (Eds.). Springer International Publishing, Cham, 83–98. https://doi.org/10.1007/

978-3-319-39696-5_6

[22] Wei Dou, Domenico Bianculli, and Lionel Briand. 2014. OCLR: A More Expressive, Pattern-Based Temporal Extension of OCL. In *Modelling Foundations and Applications*, Jordi Cabot and Julia Rubin (Eds.). Springer International Publishing, Cham, 51–66.

[23] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1998. Property Specification Patterns for Finite-state Verification. In *Proceedings of the Second Workshop on Formal Methods in Software Practice (FMSP '98)*. ACM, New York, NY, USA, 7–15. https://doi.org/10.1145/298595.298598

[24] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. Patterns in Property Specifications for Finite-state Verification. In *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. ACM, New York, NY, USA, 411–420. https://doi.org/10.1145/302405.302672

[25] Amal Elgammal, Oktay Turetken, Willem-Jan van den Heuvel, and Mike Papazoglou. 2016. Formalizing and applying compliance patterns for business process compliance. *Software & Systems Modeling* 15, 1 (2016), 119–146. https://doi.org/10.1007/s10270-014-0395-3

[26] K. Anders Ericsson and Herbert A. Simon. 1984. *Protocol Analysis; Verbal Reports as Data.* Bradford books/MIT Press, Cambridge, MA.

[27] EsperTech Inc. 2017. EPL Reference. http://www.espertech.com/esper/release-6.0.1/esper-reference/html/event_patterns.html. Last accessed: January 16, 2019.

[28] Yliès Falcone, Mohamad Jaber, Thanh-Hung Nguyen, Marius Bozga, and Saddek Bensalem. 2015. Runtime Verification of Component-based Systems in the BIP Framework with Formally-proved Sound and Complete Instrumentation. *Softw. Syst. Model.* 14, 1 (Feb. 2015), 173–199. https://doi.org/10.1007/s10270-013-0323-y

[29] Janet Feigenspan, Christian Kästner, Sven Apel, Jörg Liebig, Michael Schulze, Raimund Dachselt, Maria Papendieck, Thomas Leich, and Gunter Saake. 2013. Do background colors improve program comprehension in the #ifdef hell? *Empirical Software Engineering* 18, 4 (2013), 699–745.

[30] Kathrin Figl and Jan Recker. 2016. Exploring cognitive style and task-specific preferences for process representations. *Requirements Engineering* 21, 1 (01 Mar 2016), 63–85. https://doi.org/10.1007/s00766-014-0210-2

[31] John Fox and Sanford Weisberg. 2011. *An R Companion to Applied Regression* (second ed.). Sage, Thousand Oaks CA. http://socserv.socsci.mcmaster.ca/jfox/Books/Companion

[32] Stijn Goedertier, Jan Vanthienen, and Filip Caron. 2015. Declarative business process modelling: principles and modelling languages. *Enterprise Information Systems* 9, 2 (2015), 161–185. https://doi.org/10.1080/17517575.2013.830340

[33] Cornelia Haisjackl and Stefan Zugal. 2014. Investigating Differences between Graphical and Textual Declarative Process Models. In *Advanced Information Systems Engineering Workshops: CAiSE '14 International Workshops, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, Lazaros Iliadis, Michael Papazoglou, and Klaus Pohl (Eds.). Springer International Publishing, Cham, 194–206. https://doi.org/10.1007/978-3-319-07869-4_17

[34] Cornelia Haisjackl, Stefan Zugal, Pnina Soffer, Irit Hadar, Manfred Reichert, Jakob Pinggera, and Barbara Weber. 2013. Making Sense of Declarative Process Models: Common Strategies and Typical Pitfalls. In *Enterprise, Business-Process and Information Systems Modeling: 14th International Conference, BPMDS '13, Valencia, Spain, June 17-18, 2013. Proceedings*, Selmin Nurcan, Henderik A. Proper, Pnina Soffer, John Krogstie, Rainer Schmidt, Terry Halpin, and Ilia Bider (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2–17. https://doi.org/10.1007/978-3-642-38484-4_2

[35] J. Hatcliff, Xinghua Deng, M. B. Dwyer, G. Jung, and V. P. Ranganath. 2003. Cadena: An Integrated Development, Analysis, and Verification Environment for Component-Based Systems. In *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. 160–172. https://doi.org/10.1109/ICSE.2003.1201197

[36] Richard Heiberger and Naomi Robbins. 2014. Design of Diverging Stacked Bar Charts for Likert Scales and Other Applications. *Journal of Statistical Software, Articles* 57, 5 (2014), 1–32. https://doi.org/10.18637/jss.v057.i05

[37] W. Heijstek, T. Kuhne, and M. R. V. Chaudron. 2011. Experimental Analysis of Textual and Graphical Representations for Software Architecture Design. In *2011 International Symposium on Empirical Software Engineering and Measurement*. 167–176. https://doi.org/10.1109/ESEM.2011.25

[38] B. Hoisl, S. Sobernig, and M. Strembeck. 2014. Comparing Three Notations for Defining Scenario-Based Model Tests: A Controlled Experiment. In *QUATIC'14*. 95–104.

[39] Gerard J. Holzmann. 1997. The Model Checker SPIN. *IEEE Transactions on Software Engineering* 23, 5 (May 1997), 279–295. https://doi.org/10.1109/32.588521

[40] Hsiu-Fang Hsieh and Sarah E. Shannon. 2005. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research* 15, 9 (2005), 1277–1288. https://doi.org/10.1177/1049732305276687

[41] Jason Bryer, Kimberly Speerschneider. 2016. likert: Analysis and Visualization Likert Items. https://CRAN.R-project.org/package=likert. Last accessed: January 16, 2019.

[42] Andreas Jedlitschka, Marcus Ciolkowski, and Dietmar Pfahl. 2008. Reporting Experiments in Software Engineering. In *Guide to Advanced Empirical Software Engineering*, Forrest Shull, Janice Singer, and Dag I. K. Sjøberg (Eds.). Springer London, London, 201–228. https://doi.org/10.1007/978-1-84800-044-5_8

[43] Yogi Joshi, Guy Martin Tchamgoue, and Sebastian Fischmeister. 2017. Runtime Verification of LTL on Lossy Traces. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, New York, NY, USA, 1379–1386. https://doi.org/10.1145/3019612.3019827

[44] Natalia Juristo and Ana M. Moreno. 2010. *Basics of Software Engineering Experimentation* (1st ed.). Springer.

[45] Barbara Kitchenham, Lech Madeyski, David Budgen, Jacky Keung, Pearl Brereton, Stuart Charters, Shirley Gibbs, and Amnart Pohthong. 2016. Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering* (2016), 1–52.

[46] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering* 28, 8 (Aug. 2002), 721–734. https://doi.org/10.1109/TSE.2002.1027796

[47] Robert Kowalski and Marek Sergot. 1986. A Logic-Based Calculus of Events. *New Generation Computing* 4, 1 (01 Mar 1986), 67–95. https://doi.org/10.1007/BF03037383

[48] Thomas Krismayer, Rick Rabiser, and Paul Grünbacher. 2017. Mining Constraints for Event-based Monitoring in Systems of Systems. In *Proceedings of the 32nd International Conference on Automated Software Engineering (ASE '17)*. IEEE Press, Piscataway, NJ, USA, 826–831.

[49] Zheng Li, Jun Han, and Yan Jin. 2005. Pattern-Based Specification and Validation of Web Services Interaction Properties. In *Service-Oriented Computing (ICSOC '05)*, Boualem Benatallah, Fabio Casati, and Paolo Traverso (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 73–86.

[50] Linh Thao Ly, Stefanie Rinderle-Ma, David Knuplesch, and Peter Dadam. 2011. Monitoring Business Process Compliance Using Compliance Rule Graphs. In *On the Move to Meaningful Internet Systems (OTM '11)*, Robert Meersman, Tharam Dillon, Pilar Herrero, Akhil Kumar, Manfred Reichert, Li Qing, Beng-Chin Ooi, Ernesto Damiani, Douglas C. Schmidt, Jules White, Manfred Hauswirth, Pascal Hitzler, and Mukesh Mohania (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 82–99.

[51] Fabrizio Maria Maggi, Michael Westergaard, Marco Montali, and Wil M. P. van der Aalst. 2012. Runtime Verification of LTL-Based Declarative Process Models. In *Runtime Verification*, Sarfraz Khurshid and Koushik Sen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 131–146.

[52] Lilian Mendes Cunha, Claudia Cappelli, and Flávia Maria Santoro. 2017. Semiotic Engineering to Define a Declarative Citizen Language. In *Human Interface and the Management of Information: Supporting Learning, Decision-Making and Collaboration: 19th International Conference, HCI International 2017, Vancouver, BC, Canada, July 9–14, 2017, Proceedings, Part II*, Sakae Yamamoto (Ed.). Springer International Publishing, Cham, 503–515. https://doi.org/10.1007/978-3-319-58524-6_40

[53] George Miller. 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. , 81–97 pages.

[54] Marco Montali. 2010. The ConDec Language. In *Specification and Verification of Declarative Open Interaction Models: A Logic-Based Approach*. Springer Berlin Heidelberg, Berlin, Heidelberg, 47–75. https://doi.org/10.1007/978-3-642-14538-4_3

[55] Marco Montali, Fabrizio M. Maggi, Federico Chesani, Paola Mello, and Wil M. P. van der Aalst. 2014. Monitoring Business Constraints with the Event Calculus. *ACM Trans. Intell. Syst. Technol.* 5, 1, Article 17 (Jan. 2014), 30 pages. https://doi.org/10.1145/2542182.2542199

[56] Jeremy Morse, Lucas Cordeiro, Denis Nicole, and Bernd Fischer. 2015. Model Checking LTL Properties over ANSI-C Programs with Bounded Traces. *Softw. Syst. Model.* 14, 1 (Feb. 2015), 65–81. https://doi.org/10.1007/s10270-013-0366-0

[57] Babak Naimi, Nicholas a.s. Hamm, Thomas A. Groen, Andrew K. Skidmore, and Albertus G. Toxopeus. 2014. Where is positional uncertainty a problem for species distribution modelling. *Ecography* 37 (2014), 191–203. https://doi.org/10.1111/j.1600-0587.2013.00205.x

[58] Kioumars Namiri and Nenad Stojanovic. 2007. Using Control Patterns in Business Processes Compliance. In *Web Information Systems Engineering – WISE '07 Workshops*, Mathias Weske, Mohand-Saïd Hacid, and Claude Godart (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 178–190.

[59] OMG. 2011. BPMN 2.0. http://www.omg.org/spec/BPMN/2.0/PDF. Last accessed: January 16, 2019.

[60] Maja Pešić, Dragan Bošnački, and Wil M. P. van der Aalst. 2010. Enacting Declarative Languages Using LTL: Avoiding Errors and Improving Performance. In *Model Checking Software*, Jaco van de Pol and Michael Weber (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 146–161.

[61] Maja Pesic, Helen Schonenberg, and Wil M. P. van der Aalst. 2007. DECLARE: Full Support for Loosely-Structured Processes. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC '07)*. IEEE Computer Society, Washington, DC, USA, 287–. http://dl.acm.org/citation.cfm?id=1317532.1318056

[62] M. Pesic and W. M. P. van der Aalst. 2006. A Declarative Approach for Flexible Business Processes Management. In *Business Process Management Workshops*, Johann Eder and Schahram Dustdar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 169–180.

[63] Peter Filzmoser and Moritz Gschwandtner. 2017. mvoutlier: Multivariate Outlier Detection Based on Robust Methods. https://CRAN.R-project.org/package=mvoutlier. Last accessed: January 16, 2019.

[64] Paul Pichler, Barbara Weber, Stefan Zugal, Jakob Pinggera, Jan Mendling, and Hajo A. Reijers. 2012. Imperative versus Declarative Process Modeling Languages: An Empirical Investigation. In *Business Process Management Workshops: BPM '11 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*, Florian Daniel, Kamel Barkaoui, and Schahram Dustdar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 383–394. https://doi.org/10.1007/978-3-642-28108-2_37

[65] Amir Pnueli. 1977. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS '77)*. IEEE Computer Society, Washington, DC, USA, 46–57. https://doi.org/10.1109/SFCS.1977.32

[66] Amalinda Post, Igor Menzel, Jochen Hoenicke, and Andreas Podelski. 2012. Automotive Behavioral Requirements Expressed in a Specification Pattern System: A Case Study at BOSCH. *Requir. Eng.* 17, 1 (March 2012), 19–33. https://doi.org/10.1007/s00766-011-0145-9

[67] H. A. Reijers and J. Mendling. 2011. A Study Into the Factors That Influence the Understandability of Business Process Models. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41, 3 (May 2011), 449–462. https://doi.org/10.1109/TSMCA.2010.2087017

[68] William Revelle. 2017. *psych: Procedures for Psychological, Psychometric, and Personality Research.* Northwestern University, Evanston, Illinois. https://CRAN.R-project.org/package=psych R package version 1.7.5.

[69] Melanie A. Revilla, Willem E. Saris, and Jon A. Krosnick. 2014. Choosing the Number of Categories in Agree-Disagree Scales. *Sociological Methods & Research* 43, 1 (2014), 73–97. https://doi.org/10.1177/0049124113509605

[70] J. J. Rogmann. 2013. Ordinal Dominance Statistics (orddom): An R Project for Statistical Computing package to compute ordinal, nonparametric alternatives to mean comparison (Version 3.1). Available online from the CRAN website http://cran.r-project.org/.

[71] Marcella Rovani, Fabrizio M. Maggi, Massimiliano de Leoni, and Wil M.P. van der Aalst. 2015. Declarative Process Mining in Healthcare. *Expert Syst. Appl.* 42, 23 (Dec. 2015), 9236–9251. https://doi.org/10.1016/j.eswa.2015.07.040

[72] Kristin Y. Rozier. 2011. Survey: Linear Temporal Logic Symbolic Model Checking. *Comput. Sci. Rev.* 5, 2 (May 2011), 163–203. https://doi.org/10.1016/j.cosrev.2010.06.002

[73] Helen Schonenberg, Ronny Mans, Nick Russell, Nataliya Mulyar, and Wil van der Aalst. 2008. Process Flexibility: A Survey of Contemporary Approaches. In *Advances in Enterprise Engineering I*, Jan L. G. Dietz, Antonia Albani, and Joseph Barjis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 16–30.

[74] Z. Sharafi, A. Marchetto, A. Susi, G. Antoniol, and Y. G. Guéhéneuc. 2013. An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In *21st International Conference on Program Comprehension (ICPC '13)*. 33–42. https://doi.org/10.1109/ICPC.2013.6613831

[75] Slawomir Jarek. 2012. mvnormtest: Normality test for multivariate variables. https://CRAN.R-project.org/package=mvnormtest. Last accessed: January 16, 2019.

[76] Rachel L. Smith, George S. Avrunin, Lori A. Clarke, and Leon J. Osterweil. 2002. PROPEL: An Approach Supporting Property Elucidation. In *Proceedings of the 24th International Conference on Software Engineering (ICSE '02)*. ACM, New York, NY, USA, 11–21. https://doi.org/10.1145/581339.581345

[77] Thanh Tran, Erhard Weiss, Alexander Adensamer, Christoph Ruhsam, Christoph Czepa, Huy Tran, and Uwe Zdun. 2016. An Ontology-Based Approach for Defining Compliance Rules by Knowledge Workers in Adaptive Case Management. In *5th International Workshop on Adaptive Case Management and other Non-workflow Approaches to BPM (AdaptiveCM '16), 20th IEEE International Enterprise Computing Workshops (EDOCW '16)*. http://eprints.cs.univie.ac.at/4753/

[78] Thanh Tran, Erhard Weiss, Christoph Ruhsam, Christoph Czepa, Huy Tran, and Uwe Zdun. 2015. Embracing Process Compliance and Flexibility through Behavioral Consistency Checking in ACM: A Repair Service Management Case. In *4th International Workshop on Adaptive Case Management and other Non-workflow Approaches to BPM (AdaptiveCM '15) (Business Process Management Workshops 2015)*. http://eprints.cs.univie.ac.at/4409/

[79] Thanh Tran, Erhard Weiss, Christoph Ruhsam, Christoph Czepa, Huy Tran, and Uwe Zdun. 2015. Enabling Flexibility of Business Processes by Compliance Rules: A Case Study from the Insurance Industry. In *13th International Conference on Business Process Management (BPM '15)*. http://eprints.cs.univie.ac.at/4399/

[80] W. M. P. van der Aalst and M. Pesic. 2006. DecSerFlow: Towards a Truly Declarative Service Flow Language. In *Web Services and Formal Methods: Third International Workshop, WS-FM '06, Vienna, Austria, September 8-9, 2006 Proceedings*, Mario Bravetti, Manuel Núñez, and Gianluigi Zavattaro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–23. https://doi.org/10.1007/11841197_1

[81] Wil M. P. van der Aalst, Michael Westergaard, and Hajo A. Reijers. 2013. Beautiful Workflows: A Matter of Taste? In *The Beauty of Functional Code: Essays Dedicated to Rinus Plasmeijer on the Occasion of His 61st Birthday*, Peter Achten and Pieter Koopman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 211–233. https://doi.org/10.1007/978-3-642-40355-2_15

[82] Barbara Weber, Hajo A. Reijers, Stefan Zugal, and Werner Wild. 2009. The Declarative Approach to Business Process Execution: An Empirical Test. In *Advanced Information Systems Engineering: 21st International Conference, CAiSE '09, Amsterdam, The Netherlands, June 8-12, 2009. Proceedings*, Pascal van Eck, Jaap Gordijn, and Roel Wieringa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 470–485. https://doi.org/10.1007/978-3-642-02144-2_37

[83] Hadley Wickham. 2009. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org

[84] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.

[85] Peter Y. H. Wong and Jeremy Gibbons. 2009. Property Specifications for Workflow Modelling. In *Integrated Formal Methods*, Michael Leuschel and Heike Wehrheim (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 56–71.

[86] Jian Yu, Tan Phan Manh, Jun Han, Yan Jin, Yanbo Han, and Jianwu Wang. 2006. Pattern Based Property Specification and Verification for Service Composition. In *Web Information Systems – WISE '06: 7th International Conference on Web Information Systems Engineering, Wuhan, China, October 23-26, 2006. Proceedings*, Karl Aberer, Zhiyong Peng, Elke A. Rundensteiner, Yanchun Zhang, and Xuhui Li (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 156–168. https://doi.org/10.1007/11912873_18

[87] Stefan Zugal, Pnina Soffer, Cornelia Haisjackl, Jakob Pinggera, Manfred Reichert, and Barbara Weber. 2015. Investigating expressiveness and understandability of hierarchy in declarative business process models. *Software & Systems Modeling* 14, 3 (01 Jul 2015), 1081–1103. https://doi.org/10.1007/s10270-013-0356-2

Received ; revised ; accepted