On the Understandability of Temporal Properties Formalized in Linear Temporal Logic, Property Specification Patterns and Event Processing Language

Christoph Czepa and Uwe Zdun

Abstract—Temporal properties are important in a wide variety of domains for different purposes. For example, they can be used to avoid architectural drift in software engineering or to support the regulatory compliance of business processes. In this work, we study the understandability of three major temporal property representations: (1) Linear Temporal Logic (LTL) is a formal and well-established logic that offers temporal operators to describe temporal properties; (2) Property Specification Patterns (PSP) are a collection of recurring temporal properties that abstract underlying formal and technical representations; (3) Event Processing Language (EPL) can be used for runtime monitoring of event streams using Complex Event Processing. We conducted two controlled experiments with 216 participants in total to study the understandability of those approaches using a completely randomized design with one alternative per experimental unit. We hypothesized that PSP, as a highly abstracting pattern language, is easier to understand than LTL and EPL, and that EPL, due to separation of concerns (as one or more queries can be used to explicitly define the truth value change that an observed event pattern causes), is easier to understand than LTL. We found evidence supporting our hypotheses which was statistically significant and reproducible.

Index Terms—Controlled Experiment, Understandability, Temporal Property, Linear Temporal Logic, Property Specification Patterns, Complex Event Processing, Event Processing Language

1 Introduction

Temporal properties focus on the execution of a system, which usually involves changing states at different points in time during system execution. They play a major role in many domains, such as satellite systems (cf. Esteve et al. [1]), health care (cf. Rovani et al. [2]), banking (cf. Bianculli et al. [3]), automotive (cf. Post et al. [4]), to name a few. They are used in the context of verification and validation activities, both at design time (cf. Kherbouche et al. [5], Czepa et al. [6], Morimoto [7], and Bucchiarone et al. [8]) and at run time (cf. Mulo et al. [9], Knuplesch et al. [10], Ly et al. [11], and de Silva & Balasubramaniam [12]).

In this study, we consider a representative set of established approaches for the specification of temporal properties, namely:

- Linear Temporal Logic (LTL; cf. Pnueli [13]),
- Property Specification Patterns (PSP; cf. Dwyer et al. [14]), and
- Event Processing Language (EPL; cf. EsperTech Inc. [15]).

Linear Temporal Logic (LTL) is a widely used and established language for the specification of temporal properties. It is a logic-based approach that supports not only logical but also temporal operators. Many existing model checkers leverage LTL as a specification language (cf. Cimatti et al. [16] for NuSMV[1], Blom et al. [17] for LTSmin[2], Holzmann [18] for SPIN[3]). Originally developed for reasoning on infinite traces, LTL can also be applied for reasoning on finite traces (cf. De Giacomo & Vardi [19]). The `LTL2NFA` algorithm (cf. De Giacomo et al. [20]) describes the transformation of an arbitrary LTL formula to a non-deterministic finite automaton (NFA), which can be executed for runtime checking of LTL-based temporal properties.

The Property Specification Patterns (PSP) are a collection of recurring temporal patterns. The relevance of the patterns discovered by Dwyer et al. [14] was confirmed even 13 years after the original study took place by a survey by Bianculli et al. [3] based on 104 scientific case studies. Each pattern represents a specific intent with a mapping to underlying formal representations, most notably LTL and CTL (Computation Tree Logic; cf. Clarke et al. [21]). Many existing approaches reuse PSP or extend the original pattern catalog with more specific context-dependent patterns. Among them are the DecSerFlow language for declarative service descriptions (cf. van der Aalst & Pesic [22]), the declarative workflow approach Declare (cf. Pesic et al. [23]), the Compliance Request Language (abbrev. CRL; cf. Elgammal et al. [24]), and the PROPOLS approach for the verification of BPEL service composition schemes (cf. Yu et al. [25]).

Event Processing Language (EPL) can be used to encode specific event patterns in queries that cause the firing of

• *The authors are with the Research Group Software Architecture, Faculty of Computer Science, University of Vienna, Währingerstraße 29, 1090 Vienna, Austria*
E-mail: christoph.czepa@univie.ac.at, uwe.zdun@univie.ac.at

1. http://nusmv.fbk.eu/
2. http://fmt.cs.utwente.nl/tools/ltsmin/
3. http://spinroot.com/

event listeners once the pattern is observed in the event stream of a Complex Event Processing (CEP) environment (cf. Wu et al. [26]). EPL is part of the open source CEP engine *Esper*[4]. Numerous studies make use of EPL (cf. Awad et al. [27], Holmes et al. [28], Boubeta-Puig et al. [29], Kunz et al. [30], Adam et al. [31], Aniello et al. [32], to name but a few). EPL is well-suited as a representative for CEP query languages as it supports common CEP query language concepts, such as *leads-to* (*sequence*, *followed-by*) and *every* (*each*) operators, that are present in many CEP query languages and engines (e.g., Siddhi[5] and TESLA [33]).

1.1 Problem Statement

Despite the long existence of many major temporal property specification approaches (e.g., Linear Temporal Logic was first proposed in 1977, the Property Specification Patterns exist since 1999), the core focus of most researchers has been on the formal/technical perspective of those approaches, whereas studying the usage point of view from an empirical perspective has not drawn much attention from researchers. Indeed, we are not aware of any existing work that provides an empirical study on the understandability of different representative temporal property specification approaches. Gaining more insights into the understandability of temporal property representations is crucial for evaluating their suitability for practical use and finding potential ways for their improvement with regards to understandability.

LTL, PSP, and EPL are all powerful approaches for automated temporal property verification and validation, but very little is known about the understandability of these approaches. Intuitively, we might hypothesize that the temporal pattern-based approach PSP is more understandable than the temporal logic-based approach LTL because the former is abstracting the latter, but scientific evidence is required to back up such claims. In this article, we investigate this and similar hypotheses by applying suitable statistical methods on the gathered empirical data.

Studying the currently existing empirical research gaps in this field 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., [34]), their customers and other company representatives at conferences and workshops, we realized that industry has a huge demand for, and shows a strong interest in, temporal property specification approaches that are applicable in practice by supporting a comprehensible, fast and accurate adoption of compliance requirements as well as their automated enactment and verification. All representative temporal property specification approaches that we study in this article are well-suited for automated computer-aided checking, but BPM vendors are still often reluctant to expose their customers to such approaches, and our discussions with industry partners (see e.g. [35], [36]), that indicate uncertainty regarding how understandable temporal property representations are, are among the reasons for this.

The application of temporal property specifications for supporting software architecture compliance in the SWE & SWA domain faces a similar issue: Architecture descriptions and design decisions (cf. Medvidovic et al. [37], Zdun et al. [38]) must be documented in a comprehensible manner for different stakeholders in the software development process. Nowadays this is still often done in natural language, which cannot be directly used (i.e., without semi-automatic natural language processing; cf. Czepa et al. [39]) for automated software architecture compliance checking. By using a temporal property language for capturing architectural descriptions and decisions, we can directly leverage those architectural descriptions for automated architecture compliance checking.

Empirical research on temporal property understandability has the potential to influence practitioners in making the decision for adopting a specific existing temporal property language and in designing future industrial temporal property specification approaches. Consequently, one of the goals of this empirical study is to pave the way for industrial or practical exploitation of temporal property specification approaches.

1.2 Research Objectives

This empirical study has the objective to investigate the understandability of representative temporal property 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 temporal property representation.

We state the experimental goal using the GQM (Goal Question Metric) goal template (cf. Basili et al. [40]) as follows:

Analyze the LTL, PSP, and EPL temporal property approaches
for the purpose of their evaluation
with respect to their understandability
from the viewpoint of the novice and moderately advanced software architect, designer or developer
in the context (environment) of the Distributed System Engineering Lab and the Advanced Software Engineering Lab courses at the Faculty of Computer Science of the University of Vienna.

1.3 Context

The study consists of two controlled experiments with 216 participants in total:

- The first run was carried out with 70 computer science students who enrolled in the course "Advanced Software Engineering Lab (ASE)" (mandatory part of the master in computer science curricula) at the University of Vienna in the winter term 2015/2016.
- The second run was carried out with 92 computer science students who enrolled in the course "Distributed System Engineering Lab (DSE)" (optional part of the bachelor and master in computer science curricula) at the University of Vienna and 54 computer science students who enrolled in the course "Advanced Software Engineering Lab (ASE)" (mandatory part of the master in computer science

4. http://www.espertech.com/esper
5. https://github.com/wso2/siddhi

curricula) at the University of Vienna in the winter term 2016/2017.

Consequently, we can differentiate between DSE and ASE participants. While the former are used as proxies for novice to moderately advanced software architects, designers or developers, the latter are used as proxies for moderately advanced software architects, designers or developers. According to Kitchenham et al. [41], using students *"is not a major issue as long as you are interested in evaluating the use of a technique by novice or nonexpert software engineers. Students are the next generation of software professionals and, so, are relatively close to the population of interest"*. Besides, a number of our students work while studying and some have even some years of industry experience (cf. Electronic Appendix A.2.1). Several existing studies take it even a step further by suggesting that students can be representatives for professionals under certain circumstances (cf. Höst et al. [42], Runeson [43], Svahnberg et al. [44], and Salman et al. [45]).

1.4 Guidelines

This work follows and respects existing guidelines for conducting and reporting empirical research in software engineering: Jedlitschka et al. [46] propose guidelines and a structured approach for reporting experiments in software engineering, which had a strong influence on the general structure and contents of this article. Those guidelines integrate (among others) the "Preliminary guidelines for empirical research in software engineering" by Kitchenham et al. [41] and standard books on empirical software engineering (cf. Wohlin et al. [47], Juristo & Moreno [48]). Moreover, we considered and applied the "Robust Statistical Methods for Empirical Software Engineering" by Kitchenham et al. [49] for the statistical evaluation of the acquired data.

2 Background on Temporal Property Representations

In this section, we discuss the general properties of the temporal property representations that are the focus of this study. Readers already familiar with one (or more) of the discussed temporal property representations may consider skipping (parts of) this section.

2.1 Linear Temporal Logic (LTL)

Propositional logic is not expressive enough to describe the behavior of systems (i.e., the ordering of events in time), so the notion of temporal logic has been introduced in 1977 (cf. Pnueli [13]). In particular, a logic called Linear Temporal Logic (LTL) for reasoning over linear traces with the temporal operators $\mathcal{G}$ (or $\Box$) for "globally" and $\mathcal{F}$ (or $\Diamond$) for "finally" is proposed. Additional temporal operators are $\mathcal{U}$ for "until", $\mathcal{W}$ for "weak until", $\mathcal{R}$ for "release", and $\mathcal{X}$ (or $\circ$) for "next". $\mathcal{G}\psi$ (or $\Box\psi$) states that ψ must be true in every point in time. $\mathcal{F}\psi$ (or $\Diamond\psi$) states that ψ must be true at some future point in time. $\psi\, \mathcal{U}\, \phi$ states that ψ remains true at least until the point in time when ϕ becomes true. $\psi\, \mathcal{R}\, \phi$ states that ψ remains true at least until and including the point in time when ϕ becomes true. $\mathcal{X}\psi$ (or $\circ\psi$) states that ψ must be true at the next point in time. LTL formulas are composed of the aforementioned temporal operators, atomic propositions (the set AP), and the boolean operators $\wedge$ (for "and"), $\vee$ for "or", $\neg$ for "not", $\rightarrow$ for "implies" (cf. Baier & Katoen [50]). The weak-until operator $\psi\, \mathcal{W}\, \phi$ is defined as $(\mathcal{G}\, \psi) \vee (\psi\, \mathcal{U}\, \phi)$.

An LTL formula is inductively defined as follows: For every $a \in AP$, a is an LTL formula. If ψ and ϕ are LTL formulas, then so are $\mathcal{G}\psi$ (or $\Box\psi$), $\mathcal{F}\psi$ (or $\Diamond\psi$), $\psi\, \mathcal{U}\, \phi$, $\psi\, \mathcal{R}\, \phi$, $\mathcal{X}\psi$ (or $\circ\psi$), $\psi \wedge \phi$, $\psi \vee \phi$, and $\neg\psi$.

The semantics of LTL over infinite traces is defined as follows: LTL formulas are interpreted as infinite words over the alphabet 2^{AP} (i.e., the alphabet are all possible propositional interpretations of the propositional symbols in AP). $\pi(i)$ denotes that state of the trace π at time instant i. We define $\pi, i \vDash \psi$ (i.e., a trace π at time instant i satisfies the LTL formula ψ) as follows:

- $\pi, i \vDash a$, for $a \in AP$ iff $a \in \pi(i)$.
- $\pi, i \vDash \neg\psi$ iff $\pi, i \nvDash \psi$.
- $\pi, i \vDash \psi \wedge \phi$ iff $\pi, i \vDash \psi$ and $\pi, i \vDash \phi$.
- $\pi, i \vDash \psi \vee \phi$ iff $\pi, i \vDash \psi$ or $\pi, i \vDash \phi$.
- $\pi, i \vDash \mathcal{X}\psi$ iff $\pi, i+1 \vDash \psi$.
- $\pi, i \vDash \mathcal{F}\psi$ iff $\exists j \geq i$, such that $\pi, j \vDash \psi$.
- $\pi, i \vDash \mathcal{G}\psi$ iff $\forall j \geq i$, such that $\pi, j \vDash \psi$.
- $\pi, i \vDash \psi\, \mathcal{U}\, \phi$ iff $\exists j \geq i$, such that $\pi, j \vDash \phi$, and $\forall k, i \leq k < j$, we have $\pi, k \vDash \psi$.
- $\pi, i \vDash \psi\, \mathcal{R}\, \phi$ iff $\forall j \geq i$, iff $\pi, j \nvDash \phi$, then $\exists k, i \leq k < j$, such that $\pi, k \vDash \psi$.

In model checking, LTL formulas commonly have two possible truth value states, namely `true` (`satisfied`) and `false` (`violated`). In case of monitoring an LTL specification in a running system, 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 is 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 LTL in the experiment, we employ the semantics of Runtime Verification Linear Temporal Logic (RV-LTL; cf. Bauer et al. [51]) that supports four truth value states. In particular, an LTL temporal property specification at runtime is either `temporarily satisfied`, `temporarily violated`, `permanently satisfied`, or `permanently violated`.

The semantics of RV-LTL is defined as follows:

- $[u \vDash \psi]_{RV} = \top$ (ψ permanently satisfied by u) if for each possible finite continuation v of $u : uv \vDash \psi$.
- $[u \vDash \psi]_{RV} = \bot$ (ψ permanently violated by u) if for each possible finite continuation v of $u : uv \nvDash \psi$.
- $[u \vDash \psi]_{RV} = \top^p$ (ψ possibly/temporarily satisfied by u) if $u \vDash \psi$ and there exists a possible finite continuation v of $u : uv \nvDash \psi$.
- $[u \vDash \psi]_{RV} = \bot^p$ (ψ possibly/temporarily violated by u) if $u \nvDash \psi$ and there exists a possible finite continuation v of $u : uv \vDash \psi$.

Several existing studies make use of the concept of four LTL truth value states (cf. Pešić et al. [52], De Giacomo et

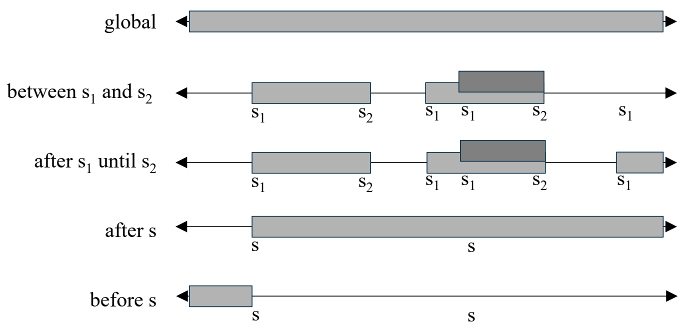

Fig. 1. Available scopes for Property Specification Patterns (shaded areas indicate the extent over which the pattern must hold)

al. [53], Maggi et al. [54], Falcone et al. [55], Joshi et al. [56], and Morse et al. [57]).

2.2 Property Specification Patterns (PSP)

Having been inspired by software design patterns, Dwyer et al. have proposed the Property Specification Patterns (PSP) [14], a collection of recurring temporal properties in software engineering. For each pattern, there exist transformation rules to underlying formal representations (including LTL and CTL)[6]. The patterns are categorized into *Occurrence Patterns* and *Order Patterns* as follows:

- 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
  - 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`)`

Moreover, 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. This scope is implicitly assumed when no other scope is defined.
- 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).

6. http://patterns.projects.cs.ksu.edu/documentation/patterns.shtml

2.3 Event Processing Language (EPL)

In this section, we discuss the Event Processing Language (EPL; cf. EsperTech Inc. [15]) and how it can be applied for runtime monitoring of temporal properties. An EPL-based temporal property specification consists of an initial truth value (either `temporarily satisfied` or `temporarily violated`) and one or more query-listener pairs. A query-listener pair causes a truth value change of the temporal property as soon as a matching event pattern is observed in the event stream. Consequently, an EPL-based temporal property specification always consists of EPL queries that are composed of EPL operators and listeners that causes truth value changes (to `temporarily satisfied`, `temporarily violated`, `permanently satisfied`, `permanently violated`) to which the state of the temporal property specification is set to by a positive match of an expression in the event stream. The semantics of those EPL operators is given as follows (cf. [15]):

- The *and* operator e_1 `and` e_2 is a logical conjunction that is matched once both e_1 and e_2 (in any order) have occurred.
- The *or* operator e_1 `or` e_2 is a logical disjunction that is matched once either e_1 or e_2 has occurred.
- The *not* operator `not` e is a logical negation that is matched if the expression e is not matched.
- The *every* operator `every` e not just observes the first occurrence of the expression e in the event stream but also each subsequent one.
- The *leads-to* operator e_1 `->` e_2 specifies that first e_1 must be observed and only then is e_2 matched. Intuitively, the whole expression is matched once e_1 is followed by e_2 at the occurrence of e_2.
- The *until* operator e_1 `until` e_2 matches the expression e_1 until e_2 occurs. In practice, this operator is commonly used in the expression `not` e_1 `until` e_2 that demands the absence of e_1 before the occurrence of e_2.

Obviously, further truth value changes are not possible once a permanent state (i.e., `permanently violated` or `permanently satisfied`) has been reached.

3 Experiment Planning

3.1 Goals

This experiment has the goal of measuring the construct *understandability* of temporal property specifications expressed in different representations, namely Linear Temporal Logic (LTL), Property Specification Patterns (PSP), and Event Processing Language (EPL). The focus is on the *correctness* and *response time* of the answers given by the participants.

3.2 Experimental Units

All participants of the experiment are students of the Faculty of Computer Science at the University of Vienna, Austria, who enrolled in the courses "Distributed System Engineering Lab (DSE)" and "Advanced Software Engineering Lab (ASE)". We differentiate between two kinds of participants:

- Participants of DSE are used as proxies for novice to moderately advanced software architects, designers or developers.
- Participants of ASE are used as proxies for moderately advanced software architects, designers or developers.

The first experiment run aims to evaluate the languages with moderately advanced software architects, designers or developers, whereas the second experiment run considers both novice to moderately advanced and moderately advanced software architects, designers or developers. Another difference between the two experiment runs concerns the incentive for participation, the sampling strategy, and the setting. In the first experiment run, the experiment was carried out as a normal course assignment. Consequently, attendance was mandatory, and the submitted solutions were graded as an integral part of the course with up to 10 points (10% of the total course points). In the second experiment run, we changed to optional attendance that was rewarded by up to 10 bonus points. In both cases, the participants' performance in the experiment determined the achieved points, and the participants were randomly allocated to the treatments (i.e., the three temporal property representations).

3.3 Experimental Material & Tasks

The temporal property specifications used in the tasks of this empirical study are based on recurring temporal property specification patterns (cf. Dwyer et al. [14] and Bianculli et al. [3]). Each task of the experiment consists of a temporal property definition and six combinations of an execution trace and a truth value. To optimize the execution of the experiment and to be independent from a specific application domain, the traces only consist of capital letters that represent surrogates of events (e.g. capital letter *A* could represent a task event *"Apply for Loan started"* in the BPM domain or a function/method invocation event in the SWA & SWE domain). For each combination the participant must evaluate whether it is correct or incorrect (i.e., whether the truth value is correct for the given trace). For example, Figure 2 (a) shows a task of the PSP group that is concerned with the *Precedence* pattern in the *Between* scope. In this task, only the choices *b)* and *f)* are correct. The same task is shown for the LTL group in Figure 2 (b) and for the EPL group in Figure 2 (c). Obviously, the expression of the temporal property in each case is changed to the appropriate formalism. Furthermore, a different set of letters is used as a preventive measure against cheating (in addition to the seating arrangements).

The experiment document consisted of 10 tasks in the first experiment run. We reduced the number of tasks in the second experiment run to 9 tasks because a relatively large number of participants could not complete the first experiment run in time. Another difference between the two experiment runs is the order of tasks and answer choices. In the first run, the order was randomized between the groups whereas in the second run there has been no difference in order between the groups. Randomization has the advantage that cheating is hampered, but it might introduce an unwanted variable to the experiment. For example, one group might have an easy first task, while another group has a hard one that hinders further progression and/or frustrates the participant. To avoid such unwanted effects, we kept the order unchanged in the second experiment run.

For the creation of the tasks of the experiment, we used an algorithm that generates traces and computes the correct truth value of a temporal property specification that corresponds to each trace automatically. This algorithm leverages both the LTL and EPL specifications used in this experiment. For checking a trace against an LTL specification, the LTL formula is transformed to a non-deterministic finite automaton (cf. De Giacomo & Vardi [19]). By executing the automaton and analyzing its accepting states, the truth value of the LTL formula can be determined. Moreover, EPL temporal property specifications are enacted in a CEP

Please select the correct answer(s) for the following constraint description:
between Z and M [T precedes S]

a) At the end of trace ***[S, T, M, M, T, T, M, M, T, L]*** the truth value of the constraint is ***permanently violated***.
b) At the end of trace ***[T, M, S, T, Z, L, L, Z, Z, L]*** the truth value of the constraint is ***temporarily satisfied***.
c) At the end of trace ***[S, L, S, M, Z, L, T, L, L, Z]*** the truth value of the constraint is ***permanently violated***.
d) At the end of trace ***[L, L, Z, Z, L, T, T, Z, S, M]*** the truth value of the constraint is ***temporarily satisfied***.
e) At the end of trace ***[Z, S, L, Z, S, M, M, M, T, L]*** the truth value of the constraint is ***temporarily satisfied***.
f) At the end of trace ***[S, Z, S, M, S, M, T, Z, S, T]*** the truth value of the constraint is ***permanently violated***.

(a) PSP group

Please select the correct answer(s) for the following constraint description:
globally((V and not K and finally K) implies (not J until (H or K)))

a) At the end of trace ***[J, H, K, K, H, H, K, K, H, L]*** the truth value of the constraint is ***permanently violated***.
b) At the end of trace ***[H, K, J, H, V, L, L, V, V, L]*** the truth value of the constraint is ***temporarily satisfied***.
c) At the end of trace ***[J, L, J, K, V, L, H, L, L, V]*** the truth value of the constraint is ***permanently violated***.
d) At the end of trace ***[L, L, V, V, L, H, H, V, J, K]*** the truth value of the constraint is ***temporarily satisfied***.
e) At the end of trace ***[V, J, L, V, J, K, K, K, H, L]*** the truth value of the constraint is ***temporarily satisfied***.
f) At the end of trace ***[J, V, J, K, J, K, H, V, J, H]*** the truth value of the constraint is ***permanently violated***.

(b) LTL group

Please select the correct answer(s) for the following constraint description:
initial truth value: temporarily satisfied
permanently violated query: every(L leads-to not S and not V until M leads-to S)

a) At the end of trace ***[M, V, S, S, V, V, S, S, V, T]*** the truth value of the constraint is ***permanently violated***.
b) At the end of trace ***[V, S, M, V, L, T, T, L, L, T]*** the truth value of the constraint is ***temporarily satisfied***.
c) At the end of trace ***[M, T, M, S, L, T, V, T, T, L]*** the truth value of the constraint is ***permanently violated***.
d) At the end of trace ***[T, T, L, L, T, V, V, L, M, S]*** the truth value of the constraint is ***temporarily satisfied***.
e) At the end of trace ***[L, M, T, L, M, S, S, S, V, T]*** the truth value of the constraint is ***temporarily satisfied***.
f) At the end of trace ***[M, L, M, S, M, S, V, L, M, V]*** the truth value of the constraint is ***permanently violated***.

(c) EPL group

Fig. 2. *Precedence Between* task in the three different treatment/group variants in the second experiment run

engine to evaluate their truth value. Using either LTL or EPL would suffice to create the tasks for the experiment. Nevertheless, we used both to double check the correctness of the temporal property representations. Please note that it is not possible to use PSP specifications directly for execution (i.e., they are an abstraction of formal languages such as LTL and EPL), so they cannot be used for automated task generation. After the automated generation, we manually checked each task for correctness.

A slightly adapted version of the algorithm was used for the second experiment run. For the first run, the truth value of an answer choice was randomly altered to another truth value to create both wrong and correct answer choices. That kind of alteration might affect the results of the EPL group because the EPL approach explicitly contains truth values in its specifications. That is, some answer choices can be ruled out by matching the truth value of an answer choice against the set of possible truth values in the EPL specification. As we will discuss later (in the evaluation of the experiments in Section 5.1), apparently, these answer choices did not introduce bias that affected the EPL results positively, but they even had a negative impact on the response times in the EPL group in the first experiment run. We eliminated that threat to validity in the second experiment run by limiting random alterations of truth values in the answer choices of all groups to the set of possible truth values of a specification.

The tasks of both controlled experiment runs are available online (cf. Czepa & Zdun [58]) to support a replication of the study. In addition, code was released as open source that supports the automated generation of experiment tasks.[7]

3.4 Hypotheses, Parameters, and Variables

We hypothesized that PSP, as a highly abstract pattern language, is easier to understand than LTL and EPL, and that EPL, due to separation of concerns (as one or more queries can be used to explicitly define the truth value change that an observed event pattern causes), is easier to understand than LTL. Consequently, we formulated the following hypotheses for the two controlled experiment runs:

- $H_{0,1}$: There is no difference in terms of understandability between PSP and LTL.
- $H_{1,1}$: PSP has a higher level of understandability than LTL.

- $H_{0,2}$: There is no difference in terms of understandability between PSP and EPL.
- $H_{1,2}$: PSP has a higher level of understandability than EPL.

- $H_{0,3}$: There is no difference in terms of understandability between EPL and LTL.
- $H_{1,3}$: EPL has a higher level of understandability than LTL.

7. https://gitlab.swa.univie.ac.at/christoph.czepa/experimentgenerator/

In both runs of this controlled experiment, there are 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 10 tasks in the first experiment run / the 9 tasks in the second experiment run.

These two dependent variables are commonly used to measure the construct understandability (cf. Feigenspan et al. [59] and Hoisl et al. [60]). The independent variable (also called factor) has three treatments, namely the three temporal property representations (LTL, EPL, and PSP).

3.5 Experiment Design & Execution

We used a completely randomized design with one alternative per experimental unit, which is appropriate for the stated goal. Through this, we tried to avoid learning effects of the participants. Moreover, chances of selection bias are limited by using a computer-aided randomization for the assignment of participants to groups. The experiment is designed as a multiple-choice test for automated processing by the e-learning platform *Moodle*[8] to avoid experimenter bias in the analysis of the answers submitted. For that reason, the participants mark the answers in an answer sheet that will be scanned and evaluated automatically. In some cases, it was necessary to correct some issues (e.g., imprecise markings) manually. To further limit the chances of experimenter bias, we used the four eyes principle while performing any such manual actions.

Two weeks before each experiment run, we handed out preparation material to the participants. This material consists of two documents: a document that provided a general introduction to the temporal property language and slides that represent a kind of quick reference guide with the important aspects of the temporal property representations and further examples. The participants were allowed to use the preparation material also during the experiment session.

The preparation material is based on (informal) natural language descriptions of the approaches and practical examples of application. There are two main reasons for this design of the preparation material: Firstly, we needed to ensure that all three languages are presented by the same educational methods at a comparable level of detail to not introduce unnecessary bias into our experiment. Secondly, we tried to present the approaches in an approachable manner to the participants as suggested by numerous existing research on teaching undergraduate students in theoretical computer science, formal methods, and logic (cf. Habiballa & Kmeť [61], Knobelsdorf & Frede [62], Carew et al. [63], Spichkova [64], and Richardson & Suinn [65]).

Please note that the tasks used in the experiment were randomly generated and not taken from the learning material. However, there were similarities between the temporal properties used in some of the experiment tasks and those used in the examples discussed in the learning material, but we could not find any indication of bias introduced by these similarities in the gathered data. In particular, the number of possibly affected experiment tasks was almost

8. https://moodle.org

balanced between the groups, and the measured correctness of possibly affected tasks was overall similar to those of the remaining tasks (cf. Electronic Appendix D.1).

Since the first experiment run also involved two qualitative questions regarding all temporal property representations, we made the decision to provide the preparation materials of all three temporal property representations to every participant. That is, the participants studied all temporal property languages, and were unaware to which group they had been assigned until the start of the experiment session. However, having knowledge of all the representations could have introduced bias. For example, learning a representation could lead to a better understanding of another one, or the languages were mixed up unintentionally. As a result, we handed out preparation material for each group individually in the second experiment run.

The preparation material is available online (cf. Czepa & Zdun [58]) to support a replication of the study.

3.6 Procedure

The first experiment run had a duration of 90 minutes for working on the 10 tasks plus an additional 10 minutes for answering the two qualitative questions. The second experiment run had a total duration of 90 minutes for working on the 9 given tasks. No qualitative questions were asked in the second run. Seating arrangements were made to limit opportunity for misbehavior (i.e., cheating). At the beginning of each experiment run, the experiment material was handed out in form of printed documents. Furthermore, we provided copies of the preparation materials for those participants who did not bring their own. Next, the participants were informed about the procedure of the experiment. This involves time tracking and how to mark answers correctly in the answer sheet for automatic processing. Following this, the participant had to fill out a general question sheet by which we gathered information about the previous knowledge and experience of the participants. Next, the main part of the experiment started, in which the participants tried to solve the tasks of the experiment. The experiment runs were carried out following this plan without known deviations.

TABLE 1
Number of observations, central tendency and dispersion per group and experiment run

		LTL	**PSP**	**EPL**
1st run	Number of observations	26	20	24
	Mean correctness [%]	33.04	69.55	50.70
	Standard deviation [%]	15.39	25.46	28.52
	Median correctness [%]	31.3	78	48.7
	Median absolute deviation [%]	12.79	23.87	42.48
	Min. correctness [%]	5	12.7	10.5
	Max. correctness [%]	63	100	94.7
	Skew (correctness)	0.02	−0.56	0.01
	Kurtosis (correctness)	−0.83	−1.01	−1.61
	Mean response time [min]	69.85	58.25	72.12
	Standard deviation [min]	15.25	20.86	21.47
	Median response time [min]	73	57.50	78.5
	Median absolute deviation [min]	17.05	25.95	17.05
	Min. response time [min]	35	28	11
	Max. response time [min]	90	90	90
2nd run: DSE	Number of observations	31	27	28
	Mean correctness [%]	32.45	70.55	53.83
	Standard deviation [%]	17.23	20.89	23.04
	Median correctness [%]	31.7	73.70	54.10
	Median absolute deviation [%]	18.09	18.09	23.5
	Min. correctness [%]	6.5	16.30	5.6
	Max. correctness [%]	70.6	97.20	86.70
	Skew (correctness)	0.36	−0.87	−0.37
	Kurtosis (correctness)	−0.62	−0.11	−0.86
	Mean response time [min]	51.03	36.65	43.80
	Standard deviation [min]	14.95	14.18	14.71
	Median response time [min]	51	33.05	42.76
	Median absolute deviation [min]	13.42	15.25	13.2
	Min. response time [min]	19	17.35	23
	Max. response time [min]	88	63.08	84.63
2nd run: ASE	Number of observations	16	17	17
	Mean correctness [%]	36.42	72.41	54.4
	Standard deviation [%]	17.32	18.17	21.06
	Median correctness [%]	38.60	71.9	53.70
	Median absolute deviation [%]	9.71	18.09	17.35
	Min. correctness [%]	3.7	33.50	8.9
	Max. correctness [%]	67.6	100	87.6
	Mean response time [min]	55.32	39.12	44
	Standard deviation [min]	11.51	8.95	15.33
	Median response time [min]	53.15	39.5	44.83
	Median absolute deviation [min]	11.48	9.64	19.74
	Min. response time [min]	35.5	23.47	23
	Max. response time [min]	78	52.93	70.5

4 Analysis

Table 1 contains the number of observations, central tendency measures and dispersion measures of the dependent variables (correctness and response time) per temporal property representation and experiment run. The second experiment run consists of measurements in two courses, namely DSE and ASE (cf. Section 3.2). That is, we tested our hypotheses three times, namely in the first experiment run in ASE, and in the second experiment run in DSE and ASE. In all three cases, the PSP group reached the highest mean and median correctness (about 70–75%), followed by the EPL group (about 50–55% correctness) and the LTL group (about 30–35% correctness). The maximum measured response time in the first run is the 90 minutes limit in all groups. In response to this, we reduced the number of tasks in the second run by one (from 10 to 9). In the second run, the maximum response time is 88 minutes. Interestingly, students in the second run in ASE managed to finish on the average about 20–40% faster than their colleagues in the first run which cannot be caused by the removal of a single task alone as the expected response time reduction would be only about 10%. We suspect that this difference is caused by the change from total experiment time recordings in the first experiment run to per task time recordings in the second experiment run, and the late assignment of participants to groups at the beginning of the experiment session in the first run. Obviously, the time recordings of the participants in the first experiment run included times such as pauses, task switching times, and times spent on consulting the accompanying documents that are not directly related to solving a specific task. In the first experiment run the participants had to be prepared for all three representations, and the experiment group was assigned at the beginning at the experiment session. Up to this point in time, the participants did not know to which experiment group they were assigned to. That is, once it became clear which of the three approaches must be applied, the participants revisited the learning material related to the assigned representation intensely. In the second experiment run, group assignment was clear beforehand, so this initial consulting of the info material did not take place in a comparable intensity. Furthermore, the mean (72.12 minutes) and median response times (78.5 minutes) of the EPL group are longer than those of the LTL group (69.85 minutes mean and 73 minutes median) in the first run. With regard to the hypotheses of this experiment, the response time measurements in the first experiment run are an unexpected result since we expected that the response times in the EPL group would be faster than in the LTL group. In contrast, the EPL group has a faster response time than the LTL group in the second run. We suspect that this effect could have been caused by the task design which contained truth value states in the answer choices that are not part of the EPL temporal property definition. Originally (i.e., at the time the first run was completed, and before the second run was carried out), we thought that there might have been a bias present in the first experiment run in favor of the EPL group, because wrong answer choices could have been potentially easier to identify by the EPL participants. However, these answer choices seemingly rather confused the participants than helped them. During the the first experiment run, EPL participants repeatedly asked whether there is an error in the exercise or whether it can be really that easy to solve it. Due to their confusion, EPL participants spent considerable more time on solving the tasks in the first experiment run. For a more detailed descriptive statistics of the dependent variables, we refer the interested reader to Electronic Appendix A.2.

After a thorough evaluation of model assumptions (cf. Electronic Appendix B), we decided to use *Cliff's delta* (cf. Cliff [66] and Rogmann [67]), a robust non-parametric test that is unaffected by change in distribution, non-normal-data and possible non-stable variance. The results of the test are shown in Table 2 for the first experiment run and Table 3 for the second experiment run. We consider FDR (False Discovery Rate) adjusted p-values (cf. Benjamini & Hochberg [68]) due to multiple testing. According to these FDR adjusted p-values, there is evidence for the rejection of the null hypotheses of this study (cf. Section 3.4).

In the first experiment run (cf. Table 2), almost all test results are significant which suggests a rejection of $H_{0,1}$ and $H_{0,2}$. $H_{0,3}$ can only be rejected on basis of the correctness variable since the test result does not indicate any significant difference in the response times of the EPL and LTL group. Moreover, the results suggest that the difference in terms of correctness between the PSP and LTL group are highly significant with a large effect size magnitude. All remaining significant test results of the first experiment run show a medium-sized effect.

In the second experiment run (cf. Table 3), the majority of the test results is significant. Only one test, namely the PSP/EPL response time with ASE participants, has no significant result, which means that $H_{0,2}$ (in ASE) can only be rejected on basis of the correctness result. All other test results are ranging from significant ($\alpha = 0.05$) to highly

TABLE 2
Cliff's d (first experiment run), one-tailed with confidence intervals calculated for $\alpha = 0.05$ (cf. Cliff [66] and Rogmann [67]), adjusted p-values (cf. Benjamini & Hochberg [68]) [Level of significance: * for $\alpha = 0.05$, ** for $\alpha = 0.01$, *** for $\alpha = 0.001$], and effect size magnitudes (cf. Kitchenham et al. [49])

		PSP/LTL	**PSP/EPL**	**EPL/LTL**
Correctness	$p_1 = P(X > Y)$	0.8769	0.7021	0.6715
	$p_2 = P(X = Y)$	0	0.0042	0
	$p_3 = P(X < Y)$	0.1231	0.2938	0.3285
	d	−0.7539	−0.4083	−0.343
	s_d	0.1097	0.1575	0.162
	z	−6.8699	−2.5933	−2.1157
	CI low	−0.8847	−0.633	−0.5789
	CI high	−0.513	−0.1203	−0.0539
	p	8.8×10^{-9}	0.0065	0.0198
	FDR adjusted p	5.3×10^{-8}	0.0195	0.0297
	level of significance	***	*	*
	effect size magnitude	large	medium	medium
Response Time	$p_1 = P(X > Y)$	0.3115	0.2833	0.564
	$p_2 = P(X = Y)$	0.0442	0.0417	0.0577
	$p_3 = P(X < Y)$	0.6442	0.675	0.3782
	d	0.3327	0.3917	−0.1859
	s_d	0.1693	0.1641	0.164
	z	1.9649	2.3874	−1.1336
	CI low	0.0312	0.0931	−0.4376
	CI high	0.5787	0.6256	0.0928
	p	0.0279	0.0108	0.1313
	FDR adjusted p	0.0335	0.0216	0.1313
	level of significance	*	*	-
	effect size magnitude	medium	medium	-

significant ($\alpha = 0.001$) which suggests a rejection of the null hypotheses. Moreover, all significant results show a large or medium effect size magnitude. It is striking that all PSP/LTL test results are highly significant with a large-sized effect.

5 Discussion

5.1 Evaluation of Results and Implications

Most results of this study are in accordance with the initial expectations of this study, but there are some deviations that must be further discussed. In the first experiment run, $H_{0,3}$ cannot be rejected for the response time variable. We suspect that this effect could be related to the experimental tasks of the first experiment run that offered answer choices with truth value states that are not part of the EPL temporal property specification. Apparently, these answer choices caused confusion that resulted in longer response times. To avoid potential bias, the answer choices in the second experiment run included only truth value states that are mentioned in the EPL temporal property specification. In the second experiment run, $H_{0,2}$ cannot be rejected for the response time variable. In this case, we could not find any plausible interpretation other than the sample size of ASE students in the second experiment run. With 50 participants, the sample size is borderline, and we cannot rule out disturbing effects. Nevertheless, aside from that, the statistical inference shows significant results with medium to large effect size magnitudes. Consequently, the controlled experiment runs of this study clearly indicate that

- PSP specifications provide a higher level of understandability than LTL specifications,

TABLE 3
Cliff's d (second experiment run), one-tailed with confidence intervals calculated for $\alpha = 0.05$ (cf. Cliff [66] and Rogmann [67]), adjusted p-values (cf. Benjamini & Hochberg [68]) [Level of significance: * for $\alpha = 0.05$, ** for $\alpha = 0.01$, *** for $\alpha = 0.001$], and effect size magnitudes (cf. Kitchenham et al. [49])

		PSP/LTL	**PSP/EPL**	**EPL/LTL**
Correctness in DSE	$p_1 = P(X > Y)$	0.902	0.709	0.7661
	$p_2 = P(X = Y)$	0	0.0053	0.0012
	$p_3 = P(X < Y)$	0.098	0.2857	0.2327
	d	-0.8041	-0.4233	-0.5334
	s_d	0.0847	0.1388	0.1275
	z	-9.4883	-3.0494	-4.1823
	CI low	-0.9053	-0.6238	-0.7107
	CI high	-0.6163	-0.1706	-0.2924
	p	1.5×10^{-13}	0.0018	5.0×10^{-5}
	FDR adjusted p	8.9×10^{-13}	0.0027	0.0002
	level of significance	***	**	***
	effect size magnitude	large	medium	large
Response Time in DSE	$p_1 = P(X > Y)$	0.2473	0.3585	0.3502
	$p_2 = P(X = Y)$	0.0024	0.0027	0.0023
	$p_3 = P(X < Y)$	0.7503	0.6389	0.6474
	d	0.503	0.2804	0.2972
	s_d	0.1345	0.153	0.1452
	z	3.7393	1.8324	2.0474
	CI low	0.251	0.0135	0.0432
	CI high	0.6911	0.51	0.5151
	p	0.0002	0.0363	0.0226
	FDR adjusted p	0.0004	0.0363	0.0271
	level of significance	***	*	*
	effect size magnitude	large	medium	medium
Correctness in ASE	$p_1 = P(X > Y)$	0.9154	0.7405	0.7427
	$p_2 = P(X = Y)$	0	0	0.0037
	$p_3 = P(X < Y)$	0.0846	0.2595	0.2537
	d	-0.8309	-0.481	-0.489
	s_d	0.0968	0.1691	0.1761
	z	-8.5879	-2.8448	-2.777
	CI low	-0.9352	-0.7108	-0.7248
	CI high	-0.5937	-0.1585	-0.1506
	p	5.3×10^{-10}	0.0038	0.0046
	FDR adjusted p	3.2×10^{-9}	0.0069	0.0069
	level of significance	***	**	**
	effect size magnitude	large	large	large
Response Time in ASE	$p_1 = P(X > Y)$	0.125	0.4187	0.2794
	$p_2 = P(X = Y)$	0.0037	0	0.0037
	$p_3 = P(X < Y)$	0.8713	0.5813	0.7169
	d	0.7463	0.1626	0.4375
	s_d	0.1172	0.2063	0.1814
	z	6.3672	0.7883	2.4124
	CI low	0.4852	-0.1854	0.0972
	CI high	0.8852	0.4744	0.6862
	p	2.2×10^{-7}	0.2	0.011
	FDR adjusted p	6.5×10^{-7}	0.2182	0.0132
	level of significance	***	-	*
	effect size magnitude	large	-	large

- PSP specifications provide a higher level of understandability than EPL specifications, and
- EPL specifications provide a higher level of understandability than LTL specifications.

When it comes to the personal preference of the participants (cf. Electronic Appendix C), PSP seems to be the most preferred temporal property representation. This result is in accordance with the outcome of the controlled experiment runs as well. In contrast, the personal preference ranking of the EPL and LTL representations does not seem to match the results of the controlled experiment runs, since the EPL representation seems to be less popular among the participants than LTL. However, the survey on which the ranking is based must be interpreted with caution, because the sample size might not be large enough to draw valid conclusions on the basis of the data. Please note that we did not replicate the survey in the second experiment run intentionally to improve the validity of the controlled experiment in the second run (cf. Section 5.2). Moreover, the constructs "personal preference" and "understandability" might be inherently different and incomparable. In either case, this peculiarity is important to report, and it might be a possible cornerstone for further investigations in future empirical studies.

Both in terms of understandability and the personal preference of the participants, the PSP representation outperformed the other two approaches examined. The pattern-based, high-level nature of the approach seems to make it highly appealing as a temporal property representation. However, a major limitation of the approach is its inflexibility in the case where the set of available patterns does not fit the purpose. In such a case, the pattern set must be extended, i.e., the creation of underlying low-level temporal property representations is required. Both EPL and LTL are more low-level temporal property representations that can be used either as underlying temporal property representations for PSP, or to directly create temporal property specifications for automated verification. EPL supports runtime monitoring, whereas LTL can be used for both runtime monitoring by non-deterministic finite automata (cf. De Giacomo et al. [20]), and design time verification by model checking (cf. Cimatti et al. [16], Blom et al. [17], Holzmann [18]). If a temporal property representation is solely used for runtime monitoring, the study would—based on the measured understandability—imply a preference for EPL over LTL. Another scenario is conceivable as well: During the creation of new PSP patterns, easier to understand EPL temporal property specifications can be used as plausibility specifications for harder to understand LTL formulas to countercheck whether a created LTL formula contains errors (cf. Czepa et al. [69]). However, an obstacle could be the possibly low user acceptance of EPL (cf. Electronic Appendix C), which must be further investigated.

5.2 Threats to Validity

All known threats that might have an impact on the validity of the results are discussed in Electronic Appendix D.

6 Related Work

To the best of our knowledge, we are not aware of any existing empirical studies that investigate the differences in understandability of representative temporal property languages in a similar way and depth as the presented study does. However, there exist related empirical studies that evaluate representations of properties/models in software engineering. This section will focus on those studies.

The first study we would like to present in the field of software architecture and engineering is indirectly related to temporal property specifications as it focuses on architecture descriptions in general. Heijstek et al. [70] try to find out whether there are differences in understanding of textual and graphical software architecture descriptions in a controlled experiment with 47 participants. Interestingly, participants who used textual architecture descriptions performed significantly better, which suggests that textual architectural descriptions could be superior to their graphical counterparts.

An eye-tracking experiment carried out by Sharafi et al. [71] with 28 participants investigates the understandability of graphical and textual software requirement models. They observed no statistically significant difference in terms of correctness of the two approaches, but the response times of participants working with the graphical representations were slower.

Czepa et al. [39] 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 EPL 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 temporal patterns (such as the Property Specification Patterns by Dwyer et al. [14]). 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. Overall, the understandability of all three approaches is at a high level. However, the results must be interpreted with caution. 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). In contrast, the controlled experiment runs presented in this article do not focus on software architecture compliance. Instead, we try to be independent from specific areas of application to evaluate the temporal property representations in a more general context. While the software architecture compliance constraints in that study wrap only a very few patterns in high-level structured languages, the empirical study presented in this article is based on a larger, representative set of temporal property patterns, and is focuses on the formalisms' core features instead of high-level, domain-specific abstractions of them.

Hoisl et al. [60] conducted a controlled experiment on three notations for defining scenario based model tests with 20 participants. In particular, they tested a semi-structured natural language scenario notation, a diagrammatic scenario notation, and a fully-structured textual scenario notation. The authors conclude that the 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. However, the validity of the experiment is strongly limited by the small sample size and the lack of statistical hypothesis testing.

7 Conclusion and Future Work

7.1 Summary

This article reports two controlled experiments on the understandability of temporal property representations with 216 participants in total (70 in the first run and 146 in the second run). The results of the statistical evaluation suggest that PSP-based temporal property specifications are significantly easier to understand than EPL temporal property specifications, that are based on Complex Event Processing (CEP), and LTL (Linear Temporal Logic) temporal property specifications. Moreover, the results imply that EPL temporal property specifications are significantly easier to understand than LTL temporal property specifications. Despite the threats to validity listed in Electronic Appendix D, we consider the validity of our results high because of the repetition and replication by a second experiment run with two different populations, the overall large sample size, the automated generation of the tasks, the automated evaluation of the given answers, and the thorough statistical evaluation.

7.2 Impact

This study seems to support the original assumption that the pattern-based PSP approach is the most user-friendly temporal property representation for novice and moderately advanced users. Therefore, if possible (i.e., if the approach is applicable to the domain), the results suggest that the pattern-based temporal property approach should be preferred. Since many existing approaches (e.g., the Compliance Request Language CRL by Elgammal et al. [24] and the PROPOLS approach for the verification of BPEL service composition schemes by Yu et al. [25]) reuse PSP or extend the original pattern catalog (cf. Dywer et al. [14]) with more specific context-dependent patterns, there is strong evidence that the results of the study hold for these approaches as well. However, in contrast to the two other temporal property approaches tested in this study, the pattern-based approach is the most limited one in terms of its expressiveness. That is, if the set of supported patterns is incompatible with a specific requirement (e.g., a company internal policy that must be covered by the IT system), it is necessary to extend the pattern catalog. Since the pattern-based approach merely abstracts other temporal property representations (most often LTL formulas), creating new patterns always requires the creation of the underlying temporal property specifications as well. Creating those underlying temporal property specifications is considered to be difficult and error-prone. Plausibility checking (cf. Czepa et al. [69]) tries to alleviate the risk to create incorrect LTL specifications by leveraging EPL specifications to countercheck if the LTL formula contains errors. Since EPL temporal property specifications are more understandable than LTL formulas, the results of the presented study can be seen as an empirical evaluation of the plausibility checking approach as well.

7.3 Future Work

The presented study focuses on the understandability of already given temporal properties. That is, the authoring of temporal property specifications is not yet sufficiently covered. It is possible to further investigate the understandability of temporal property languages by running different kinds of experiments. In particular, we plan to study the understandability of temporal property representations during the authoring process as well. We suspect that creating correct temporal property specifications from scratch is more difficult than interpreting already given temporal property specifications correctly. Moreover, we are curious whether the measured significant differences in understandability of the three temporal property representations are also present during the creation process of temporal property specifications. Another interesting opportunity for future work is studying the understandability of temporal property specifications with professionals working in the industry (e.g., senior system administrators and senior software architects). Studying whether there exist differences in understandability between textual and graphical temporal property representation is another interesting opportunity for future work. In particular, it would be interesting to find out whether the results of the studies by Heijstek et al. [70] and Sharafi et al. [71], that investigated the differences in understandability of textual and graphical models in the software architecture and engineering domain with results in favor of the textual approaches, are transferable to temporal property specifications. In this context, it might be interesting as well to compare textual LTL representations against the graphical NFA representations since NFAs are often the transformation product of LTL formulas.

Acknowledgments

We would like to thank all participants. This work was supported by: FFG (Austrian Research Promotion Agency) project CACAO, no. 843461; FWF (Austrian Science Fund) project ADDCompliance: I 2885-N33

References

[1] M.-A. Esteve, J.-P. Katoen, V. Y. Nguyen, B. Postma, and Y. Yushtein, "Formal correctness, safety, dependability, and performance analysis of a satellite," in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 1022–1031. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337354

[2] M. Rovani, F. M. Maggi, M. de Leoni, and W. M. van der Aalst, "Declarative process mining in healthcare," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9236–9251, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2015.07.040

[3] D. Bianculli, C. Ghezzi, C. Pautasso, and P. Senti, "Specification patterns from research to industry: A case study in service-based applications," in *2012 34th International Conference on Software Engineering (ICSE)*, June 2012, pp. 968–976.

[4] A. Post, I. Menzel, J. Hoenicke, and A. Podelski, "Automotive behavioral requirements expressed in a specification pattern system: A case study at bosch," *Requir. Eng.*, vol. 17, no. 1, pp. 19–33, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1007/s00766-011-0145-9

[5] O. M. Kherbouche, A. Ahmad, and H. Basson, "Formal approach for compliance rules checking in business process models," in *2013 IEEE 9th International Conference on Emerging Technologies (ICET)*, Dec 2013, pp. 1–6.

[6] C. Czepa, H. Tran, U. Zdun, T. Tran, E. Weiss, and C. Ruhsam, "Reduction techniques for efficient behavioral model checking in adaptive case management," in *The 32nd ACM Symposium on Applied Computing (SAC 2017)*, April 2017. [Online]. Available: http://eprints.cs.univie.ac.at/4879/

[7] S. Morimoto, *A Survey of Formal Verification for Business Process Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 514–522. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-69387-15_8

[8] A. Bucchiarone, H. Muccini, P. Pelliccione, and P. Pierini, *Model-Checking Plus Testing: From Software Architecture Analysis to Code Testing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 351–365. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30233-9_26

[9] E. Mulo, U. Zdun, and S. Dustdar, "Domain-specific language for event-based compliance monitoring in process-driven soas," *Service Oriented Computing and Applications*, vol. 7, no. 1, pp. 59–73, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11761-012-0121-3

[10] D. Knuplesch, M. Reichert, L. T. Ly, A. Kumar, and S. Rinderle-Ma, "On the formal semantics of the extended compliance rule graph," Ulm University, Ulm, Technical Report UIB-2013 - 05, April 2013. [Online]. Available: http://dbis.eprints.uni-ulm.de/1147/

[11] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. van der Aalst, "Compliance monitoring in business processes," *Inf. Syst.*, vol. 54, no. C, pp. 209–234, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.is.2015.02.007

[12] L. de Silva and D. Balasubramaniam, *PANDArch: A Pluggable Automated Non-intrusive Dynamic Architecture Conformance Checker*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 240–248. [Online]. Available: https://doi.org/10.1007/978-3-642-39031-9_21

[13] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, ser. SFCS '77. Washington, DC, USA: IEEE Computer Society, 1977, pp. 46–57. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1977.32

[14] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21st International Conference on Software Engineering*, ser. ICSE '99. New York, NY, USA: ACM, 1999, pp. 411–420. [Online]. Available: http://doi.acm.org/10.1145/302405.302672

[15] EsperTech Inc., "EPL Reference," http://www.espertech.com/esper/release-6.0.1/esper-reference/html/event_patterns.html, 2017, last accessed: August 22, 2018.

[16] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in *Proceedings of the 14th International Conference on Computer Aided Verification*, ser. CAV '02. London, UK, UK: Springer-Verlag, 2002, pp. 359–364. [Online]. Available: http://dl.acm.org/citation.cfm?id=647771.734431

[17] S. Blom, J. van de Pol, and M. Weber, *LTSmin: Distributed and Symbolic Reachability*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 354–359. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14295-6_31

[18] G. J. Holzmann, "The model checker spin," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, May 1997. [Online]. Available: http://dx.doi.org/10.1109/32.588521

[19] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, pp. 854–860. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540252

[20] G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on ltl on finite traces: Insensitivity to infiniteness," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, pp. 1027–1033. [Online]. Available: http://dl.acm.org/citation.cfm?id=2893873.2894033

[21] E. M. Clarke and E. A. Emerson, *Design and synthesis of synchronization skeletons using branching time temporal logic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 52–71. [Online]. Available: http://dx.doi.org/10.1007/BFb0025774

[22] W. M. P. van der Aalst and M. Pesic, *DecSerFlow: Towards a Truly Declarative Service Flow Language*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–23. [Online]. Available: http://dx.doi.org/10.1007/11841197_1

[23] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "Declare: Full support for loosely-structured processes," in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, ser. EDOC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 287–. [Online]. Available: http://dl.acm.org/citation.cfm?id=1317532.1318056

[24] A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou, "Formalizing and appling compliance patterns for business process compliance," *Software & Systems Modeling*, vol. 15, no. 1, pp. 119–146, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10270-014-0395-3

[25] J. Yu, T. P. Manh, J. Han, Y. Jin, Y. Han, and J. Wang, *Pattern Based Property Specification and Verification for Service Composition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 156–168. [Online]. Available: http://dx.doi.org/10.1007/11912873_18

[26] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '06. New York, NY, USA: ACM, 2006, pp. 407–418. [Online]. Available: http://doi.acm.org/10.1145/1142473.1142520

[27] A. Awad, A. Barnawi, A. Elgammal, R. Elshawi, A. Almalaise, and S. Sakr, "Runtime detection of business process compliance violations: An approach based on anti patterns," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 1203–1210. [Online]. Available: http://doi.acm.org/10.1145/2695664.2699488

[28] T. Holmes, E. Mulo, U. Zdun, and S. Dustdar, *Model-aware Monitoring of SOAs for Compliance*. Vienna: Springer Vienna, 2011, pp. 117–136. [Online]. Available: https://doi.org/10.1007/978-3-7091-0415-6_5

[29] J. Boubeta-Puig, G. Daz, H. Maci, V. Valero, and G. Ortiz, "Medit4cep-cpn: An approach for complex event processing modeling by prioritized colored petri nets," *Information Systems*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437917300108

[30] S. Kunz, T. Fickinger, J. Prescher, and K. Spengler, "Managing complex event processes with business process modeling notation," in *Business Process Modeling Notation*, J. Mendling, M. Weidlich, and M. Weske, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 78–90.

[31] M. Adam, C. Cordeiro, L. Field, D. Giordano, and L. Magnoni, "Real-time complex event processing for cloud resources," *Journal of Physics: Conference Series*, vol. 898, no. 4, p. 042020, 2017.

[32] L. Aniello, G. A. Di Luna, G. Lodi, and R. Baldoni, "A collaborative event processing system for protection of critical infrastructures from cyber attacks," in *Computer Safety, Reliability, and Security*, F. Flammini, S. Bologna, and V. Vittorini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 310–323.

[33] G. Cugola and A. Margara, "Tesla: A formally defined event specification language," in *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '10. New York, NY, USA: ACM, 2010, pp. 50–61. [Online]. Available: http://doi.acm.org/10.1145/1827418.1827427

[34] T. Tran, E. Weiss, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun, "Enabling flexibility of business processes by compliance rules: A case study from the insurance industry," in *13th International Conference on Business Process Management 2015, Industry Track*, August 2015. [Online]. Available: http://eprints.cs.univie.ac.at/4399/

[35] ——, "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)*, ser. Business Process Management Workshops 2015, August 2015. [Online]. Available: http://eprints.cs.univie.ac.at/4409/

[36] T. Tran, E. Weiss, A. Adensamer, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun, "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 2016)*, September 2016. [Online]. Available: http://eprints.cs.univie.ac.at/4753/

[37] N. Medvidovic, D. S. Rosenblum, and R. N. Taylor, "A language and environment for architecture-based software development and evolution," in *Proceedings of the 21st International Conference on Software Engineering*, ser. ICSE '99. New York, NY, USA: ACM, 1999, pp. 44–53. [Online]. Available: http://doi.acm.org/10.1145/302405.302410

[38] U. Zdun, R. Capilla, H. Tran, and O. Zimmermann, "Sustainable architectural design decisions," *IEEE Software*, vol. 30, no. 6, pp. 46–53, Nov 2013.

[39] C. Czepa, H. Tran, U. Zdun, T. Tran, E. Weiss, and C. Ruhsam, "On the understandability of semantic constraints for behavioral software architecture compliance: A controlled experiment," in *IEEE International Conference on Software Architecture (ICSA 2017)*, April 2017. [Online]. Available: http://eprints.cs.univie.ac.at/5059/

[40] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering*. Wiley, 1994.

[41] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, Aug. 2002. [Online]. Available: http://dx.doi.org/10.1109/TSE.2002.1027796

[42] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—a comparative study of students and professionals in lead-time impact assessment," *Empirical Software Engineering*, vol. 5, no. 3, pp. 201–214, Nov 2000. [Online]. Available: https://doi.org/10.1023/A:1026586415054

[43] P. Runeson, "Using students as experiment subjects an analysis on graduate and freshmen student data," in *Proceedings 7th International Conference on Empirical Assessment and Evaluation in Software Engineering*, 2003, pp. 95–102.

[44] M. Svahnberg, A. Aurum, and C. Wohlin, "Using students as subjects - an empirical evaluation," in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '08. New York, NY, USA: ACM, 2008, pp. 288–290. [Online]. Available: http://doi.acm.org/10.1145/1414004.1414055

[45] I. Salman, A. T. Misirli, and N. Juristo, "Are students representatives of professionals in software engineering experiments?" in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 666–676. [Online]. Available: http://dl.acm.org/citation.cfm?id=2818754.2818836

[46] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, *Reporting Experiments in Software Engineering*. London: Springer London, 2008, pp. 201–228. [Online]. Available: http://dx.doi.org/10.1007/978-1-84800-044-5_8

[47] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[48] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*, 1st ed. Springer, 2010.

[49] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong, "Robust statistical methods for empirical software engineering," *Empirical Software Engineering*, pp. 1–52, 2016.

[50] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[51] A. Bauer, M. Leucker, and C. Schallhart, "Comparing ltl semantics for runtime verification," *J. Log. and Comput.*, vol. 20, no. 3, pp. 651–674, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1093/logcom/exn075

[52] M. Pešić, D. Bošnački, and W. M. P. van der Aalst, "Enacting declarative languages using ltl: Avoiding errors and improving performance," in *Model Checking Software*, J. van de Pol and M. Weber, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 146–161.

[53] G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, and M. Montali, "Monitoring business metaconstraints based on ltl and ldl for finite traces," in *Business Process Management*, S. Sadiq, P. Soffer, and H. Völzer, Eds. Cham: Springer International Publishing, 2014, pp. 1–17.

[54] F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst, "Runtime verification of ltl-based declarative process models," in *Runtime Verification*, S. Khurshid and K. Sen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 131–146.

[55] Y. Falcone, M. Jaber, T.-H. Nguyen, M. Bozga, and S. Bensalem, "Runtime verification of component-based systems in the bip framework with formally-proved sound and complete

instrumentation," *Softw. Syst. Model.*, vol. 14, no. 1, pp. 173–199, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10270-013-0323-y

[56] Y. Joshi, G. M. Tchamgoue, and S. Fischmeister, "Runtime verification of ltl on lossy traces," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. New York, NY, USA: ACM, 2017, pp. 1379–1386. [Online]. Available: http://doi.acm.org/10.1145/3019612.3019827

[57] J. Morse, L. Cordeiro, D. Nicole, and B. Fischer, "Model checking ltl properties over ansi-c programs with bounded traces," *Softw. Syst. Model.*, vol. 14, no. 1, pp. 65–81, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10270-013-0366-0

[58] C. Czepa and U. Zdun, "On the Understandability of Temporal Properties Formalized in Linear Temporal Logic, Property Specification Patterns and Event Processing Language [Data set]," http://doi.org/10.5281/zenodo.891007, 2017.

[59] J. Feigenspan, C. Kästner, S. Apel, J. Liebig, M. Schulze, R. Dachselt, M. Papendieck, T. Leich, and G. Saake, "Do background colors improve program comprehension in the #ifdef hell?" *Empirical Software Engineering*, vol. 18, no. 4, pp. 699–745, 2013.

[60] B. Hoisl, S. Sobernig, and M. Strembeck, "Comparing three notations for defining scenario-based model tests: A controlled experiment," in *QUATIC'14*, Sept 2014, pp. 95–104.

[61] H. Habiballa and T. Kmet, "Theoretical branches in teaching computer science," *International Journal of Mathematical Education in Science and Technology*, vol. 35, no. 6, pp. 829–841, 2004. [Online]. Available: https://doi.org/10.1080/00207390412331271267

[62] M. Knobelsdorf and C. Frede, "Analyzing student practices in theory of computation in light of distributed cognition theory," in *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ser. ICER '16. New York, NY, USA: ACM, 2016, pp. 73–81. [Online]. Available: http://doi.acm.org/10.1145/2960310.2960331

[63] D. Carew, C. Exton, and J. Buckley, "An empirical investigation of the comprehensibility of requirements specifications," in *2005 International Symposium on Empirical Software Engineering, 2005.*, Nov 2005, pp. 10 pp.–.

[64] M. Spichkova, ""boring formal methods" or "sherlock holmes deduction methods"?" in *Software Technologies: Applications and Foundations*, P. Milazzo, D. Varró, and M. Wimmer, Eds. Cham: Springer International Publishing, 2016, pp. 242–252.

[65] F. Richardson and R. M. Suinn, "The mathematics anxiety rating scale," vol. 19, pp. 551–554, 11 1972.

[66] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions," *Psychological Bulletin*, vol. 114, pp. 494–509, 1993.

[67] J. J. Rogmann, "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/, 2013.

[68] Y. Benjamini and Y. Hochberg, "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[69] C. Czepa, H. Tran, U. Zdun, T. Tran, E. Weiss, and C. Ruhsam, "Plausibility checking of formal business process specifications in linear temporal logic," in *28th International Conference on Advanced Information Systems Engineering (CAiSE'16), Forum Track*, June 2016. [Online]. Available: http://eprints.cs.univie.ac.at/4692/

[70] W. Heijstek, T. Kuhne, and M. R. V. Chaudron, "Experimental analysis of textual and graphical representations for software architecture design," in *2011 International Symposium on Empirical Software Engineering and Measurement*, Sept 2011, pp. 167–176.

[71] Z. Sharafi, A. Marchetto, A. Susi, G. Antoniol, and Y. G. Guhneuc, "An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension," in *2013 21st International Conference on Program Comprehension (ICPC)*, May 2013, pp. 33–42.

Christoph Czepa is a researcher at the Faculty of Computer Science, University of Vienna, Austria. His research areas include Business Process Management (BPM), Adaptive Case Management (ACM), software architecture, software engineering, and the application of machine learning and formal verification methods in the aforementioned domains. He received a master degree in computer science in December 2013 (with distinction). Currently, he is pursuing a PhD in computer science. Christoph has published more than 15 peer-reviewed scientific articles, and he participated in two research projects, namely the CACMTV (Content-Aware Coding for Mobile TV) project and CACAO (Consistency Checking, Recommendations, And Visual Modeling For Ad Hoc Changes By Knowledge Workers In Adaptive Case Management) project.

Uwe Zdun is a full professor for software architecture at the Faculty of Computer Science, University of Vienna. Before that, he worked as assistant professor at the Vienna University of Technology and the Vienna University of Economics respectively. He received his doctoral degree from the University of Essen in 2002. His research focuses on software design and architecture, empirical software engineering, distributed systems engineering (service-based, cloud, mobile, and process-driven systems), software patterns, domain-specific languages, and model-driven development. Uwe has published more than 210 articles in peer-reviewed journals, conferences, book chapters, and workshops, and is co-author of the books "Remoting Patterns Foundations of Enterprise, Internet, and Realtime Distributed Object Middleware", "Process-Driven SOA Proven Patterns for Business-IT Alignment, and Software-Architektur." He has participated in 26 R&D projects. Uwe is editor of the journal Transactions on Pattern Languages of Programming (TPLoP) published by Springer, Associate Editor of the Computing journal published by Springer, and Associate Editor-in-Chief for design and architecture for the IEEE Software magazine.