An Empirical Study of the Intuitive Understanding of a Formal Pattern Language

Elisabeth Henkel[0000−0003−3844−8292], Nico Hauff[0000−0002−8972−2776], Lukas Eber, Vincent Langenfeld[0000−0001−9835−6790], and Andreas Podelski[0000−0003−2540−9489]

Department of Computer Science, University of Freiburg,
`{hauffn,henkele,langenfv,podelski}@informatik.uni-freiburg.de`

Abstract. [Context and motivation] Formal pattern languages with a restricted English grammar, such as the pattern language of Konrad and Cheng, give us the possibility to combine human intuition and the rigour of a machine. **[Question / problem]** The question arises to what extent the intuitive understanding of such a pattern language is in agreement with its formal semantics. **[Principal ideas / results]** We present an empirical study to address this question. The existence of a formal semantics allows us to use the machine as an objective judge to decide if the intuitive understanding is correct. The study confirms empirically the practical usefulness of HanforPL in that the intuitive understanding matches the formal semantics in most practically relevant cases. The study reveals that a number of *phrases of interest* represent critical edge cases where even a prior exposure to formal logic is not a guarantee for the correct intuitive understanding. **[Contribution]** We show how the alignment of formal and intuitive semantics can be investigated, and that this alignment can not simply be assumed. Nonetheless, results regarding the understandability of HanforPL are favourable with high understandability in commonly used patterns. The results of the study will be the basis of improvements in HanforPL.

Keywords: Pattern Languages · Formal Requirements · Intuitive Understanding · Empirical Study.

1 Introduction

The formal representation of requirements is supposed to overcome some of the deficiencies of natural language requirements, especially lack of precision and non-machine readability [16,2,15,5]. However, if requirements are formulated in a formal logic such as temporal logic, they are accessible to only a restricted group of requirement engineers. To overcome the lack of general accessibility, Konrad and Cheng introduced a pattern language to formulate formal requirements as sentences in a restricted English grammar [8]. The intuitive understanding of these sentences is based on the intuitive understanding of natural language, while the formal semantics is derived through corresponding temporal logic formulas.

For example, we can use its formal semantics to uniquely determine that the requirement below is satisfied by the behaviour depicted in Figure 1:

> *Globally, it is always the case that if **R** holds, then **S** holds after **1** time unit.*

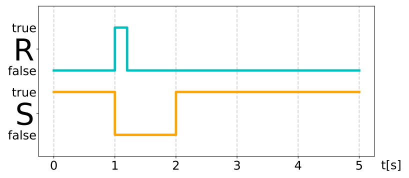


Fig. 1: Example behaviour over the observables ***R*** and ***S***.

It would thus seem that with pattern languages, we are in the ideal situation where we can have both, the precision of formal requirements and the accessibility of natural language. However, while the interface to the computer is fixed by the formal semantics, the interface to the human still relies on the intuitive interpretation of natural language. The question is to what extent we still have the issues of natural language requirements if restricted to the subset of sentences defined in a pattern language. In particular, the question arises to what extent the intuitive understanding of each requirement in the pattern language will be correct.

The existence of a formal semantics for the requirements gives us the unique opportunity to phrase the above question in a mathematically precise sense. We can give a mathematically precise definition of what is the *correct* intuitive understanding of a requirement in the pattern language, namely, through its formal meaning. In contrast, for an informal requirement, it would seem impossible to distinguish one possible intuitive understanding over another one.

For a requirement in the pattern language, the formal meaning is defined as the set of system behaviours that satisfy the corresponding temporal logic formula. Thus, we can base the test of the intuitive understanding of a requirement on a set of example behaviours, some of which satisfy the requirement and some of which do not. The existence of a formal semantics allows us to define an objective judge who decides whether the intuitive understanding is correct: the machine. Both, the requirement and the behaviour have a machine representation, and an algorithm exists to decide whether the behaviour satisfies the requirement. Thus, we only compare the intuitive understanding to the algorithmic decision.

In this paper, we report on an empirical study to investigate the difference between the formal semantics and the intuitive understanding of requirements in a particular example of a pattern language called HanforPL. The pattern language comes with a framework to specify requirements and behaviours, and to check whether a behaviour satisfies a requirement [1,5]. The study confirms empirically the practical usefulness of HanforPL in that the intuitive understanding matches the formal semantics in many cases. The study reveals that a number of *phrases of interest* represent critical edge cases where even a prior exposure to formal logic is not a guarantee for the correct intuitive understanding.

Table 1: The table shows all patterns of HanforPL, with their membership to a group describing overall behaviour (the *Order*, the *Occurrence*, or the *Real-Time*), the names of each pattern, and the pattern text. Due to the available space, we use "..." to omit the shared phrase *it is always the case that*. Names of patterns not already part of the SPS [13] are shown in blue colour.

	Name	Pattern
Order	ConstrainedChain	... if **R** holds, then **S** eventually holds and is succeeded by **T**, where **U** does not hold between **S** and **T**.
	Initialization	... initially **R** holds
	Persistence	... if **R** holds, then it holds persistently
	PrecedenceChain21	... if **R** holds, then **S** previously held and was preceded by **T**
	PrecedenceChain12	... if **R** holds and is succeeded by **S**, then **T** previously held
	Response	... if **R** holds, then **S** eventually holds.
	ResponseChain12	... if **R** holds, then **S** eventually holds and is succeeded by **T**.
	Precedence	... if **R** holds, then **S** previously held.
Occur.	Absence	it is never the case that **R** holds
	ExistenceBoundU	transitions to states in which **R** holds occur at most twice
	Invariance	... if **R** holds, then **S** holds as well
	Universality	... **R** holds
Real-time	DurationBoundL	... once **R** becomes satisfied, it holds for at least **S** time units
	DurationBoundU	... once **R** becomes satisfied, it holds for less than **S** time units
	EdgeResponseBoundU1	... once **R** becomes satisfied and holds for at most **S** time units, then **T** holds afterwards
	EdgeResponseBoundL2	... once **R** becomes satisfied, **S** holds for at least **T** time units
	EdgeResponseDelay	... once **R** becomes satisfied, **S** holds after at most **T** time units
	EdgeResponseDelayBoundL2	... once **R** becomes satisfied, **S** holds after at most **T** time units for at least **U** time units
	InvarianceBoundL2	... if **R** holds, then **S** holds for at least **T** time units
	ReccurrenceBoundL	... **R** holds at least every **S** time units
	ResponseBoundL1	... if **R** holds for at least **S** time units, then **T** holds afterwards
	ResponseBoundL12	... if **R** holds for at least **S** time units, then **T** holds afterwards for at least **U** time units
	ResponseDelay	... if **R** holds, then **S** holds after at most **T** time units
	ResponseDelayBoundL2	... if **R** holds, then **S** holds after at most **T** time units for at least **U** time units
	TriggerResponseBoundL1	... after **R** holds for at least **S** time units and **T** holds, then **U** holds
	TriggerResponseDelayBoundL1	... after **R** holds for at least **S** time units and **T** holds, then **U** holds after at most **V** time units
	UniversalityDelay	... **R** holds after at most **S** time units

2 Hanfor Pattern Language

The Hanfor pattern language (HanforPL) is based on the patterns of Konrad and Cheng [8] and uses the Duration Calculus semantics of Post [13]. In fact, HanforPL shares a large portion of patterns with the Specification Pattern System (SPS) from [13].

Each instantiation of a requirement in HanforPL is a combination of a *scope* defining the general applicability of a pattern, followed by the *pattern* itself. The scopes can be chosen from the following options *Globally*, *After* ***P***, *After* ***P*** *until* ***Q***, *Before* ***P***, and *Between* ***P*** *and* ***Q***. The resulting patterns are listed in Table 1. During instantiation placeholders (usually P, Q, R, S, T) have to be replaced by Boolean expressions over observables (using $\neg, \wedge$ for Boolean and $<, =$ for numeric observables).

The semantics of each scope and pattern combination is defined by a logical formula containing the same placeholders. For a more depth introduction to the formal foundations and the pattern semantics in detail, we kindly refer the reader to the cited work.

3 Empirical Study

In this section, we describe the overall goal of our empirical study, our research questions, and the study design.

3.1 Goal and Research Questions

As requirements pattern are used to communicate expected system behaviour, e.g., between customers or different departments, it is necessary that requirements are as understandable as possible to as many stakeholders as possible. That is, the semantics of the pattern defined by formal logics should align with the intuitive understanding of usual stakeholders.

The goal of this study is thus to investigate to what extent the intuitive understanding of formal requirements in HanforPL is correct in the sense that it matches the formal semantics. This is closely related to the question of the practical usefulness of HanforPL.

Further, we aim to identify possible reasons for misinterpretation in order to improve HanforPL in the long term.

Based on previous experience (e.g. [11,9]), we are confident that formally trained people with some training in HanforPL perform well using the pattern language. With Research Question R1, we want to investigate how well participants without any training in HanforPL understand the patterns.

However, a basic understanding of formal logics and/or requirements engineering in general may serve as a predictor for the performance dealing with edge cases and uncommon concepts (Research Question R2).

As the requirements pattern are based on natural language sentences, there may be phrases that allow for several sensible interpretations for complex concepts, e.g., formulations referring to timing constraints and quantification. These phrases of interest are investigated in detail in Research Question R3.

R1 How understandable is HanforPL without former training in the pattern language itself?

R2 Does training in the fields of requirements engineering or formal logics have a positive effect on the understanding of HanforPL patterns?

- **a)** Requirements engineering
- **b)** Formal logics

R3 How is the understanding of HanforPL impacted by complex concepts, i.e., formulations referring to timing constraints and quantification?

With regard to the last Research Question (R3), we identified several phrases used within HanforPL to describe concepts like timing constraints and quantification. In the following, we present a list of these *phrases of interest* (highlighted within the according pattern) together with a description of possible interpretations. Additionally, we state which of the possible interpretations matches the *intended meaning*, i.e., the semantic fixed by the corresponding Duration Calculus formula.

(prev) *[...] if **R** holds, then **S** previously held* : For this phrasing, we see two possible points for ambiguity. First, the phrase does not specify whether ***S*** has to hold persistently or only for a non-zero time interval before any occurrence of ***R***. And second, it is not specified whether ***S*** has to hold at an arbitrary point in time before the occurrence of ***R*** or directly before ***R*** holds. The intended meaning is the following: Every occurrence of ***R*** must at some point be preceded by a non-zero time interval in which ***S*** held.

(afterw)/(afterw*) *[...] if [...], then **S** holds afterwards* : Analogous to (prev), we identified two possible ambiguities. The phrase does not specify, whether ***S*** has to hold persistently or only for a non-zero time interval (afterw). Additionally, it is not specified, whether ***S*** has to hold directly after the trigger event (the [...]-part) or only at an arbitrary point in time after the triggered event (afterw*). The intended meaning is the following: ***S*** must hold directly after the trigger event for some non-zero time interval.

(aam) *[...] **R** holds after at most **d** seconds* : The phrase does not specify whether ***R*** has to hold persistently after the ***d*** seconds have passed (which is the intended meaning), or only has to hold for a non-zero time interval.

(aam-cond) *[...] if [...], then **S** holds after at most **d** seconds* : This wording is the conditioned version of (aam), i. e., it is dependent on the context of a preceding trigger. Analogous, it is not specified whether ***S*** has to hold persistently or only for a non-zero time interval after ***d*** seconds have passed. The intended meaning is the following: ***S*** has to hold for a non-zero time interval. However, due to an oversight while extending the pattern language, this interpretation is clearly inconsistent with the intended meaning provided in (aam).

(obs)/(obs+) *[...] once **R** becomes satisfied [...]*: We identified two possible ambiguities in this pattern. The first is regarding the meaning of the phrase *becomes satisfied*. It might be unclear, whether a rising edge of ***R*** is strictly required in all cases, or whether this phrase also includes system behaviour where ***R*** initially holds (obs). The second ambiguity concerns the keyword *once*. It might be unclear, whether this means that every occurrence of ***R*** becoming satisfied should be considered or only the first occurrence (obs+). The intended meaning is the following: all occurrences of rising edges of ***R*** should be considered.

(rec) *[...] **R** holds at least every 2 seconds* : The intended meaning of this phrase is that the length of intervals in which ***R*** does not hold is at most 2 seconds. However, this wording might be misinterpreted so to mean, that ***R*** holds at fixed points in time $t_0 = 0, t_1 = 2, t_2 = 4, \dots, t_n = 2n$.

Remark. Even though some inconsistencies, e.g., the intended meaning of *holds* in (aam) and (aam-cond), were identified while preparing the study, we decided to make no premature changes for two reasons: First, we are interested to know whether such an inconsistency is noticeable in the results. Second, if it is noticeable, which of the different interpretations is the one that most participants agree with.

3.2 Subject Selection

Participants for the empirical study were selected via convenience sampling of contacts of the authors and second-degree contacts in an original equipment manufacturer (OEM) in the automotive field. Subjects are mostly computer scientists and requirements engineers from the field of software engineering, automotive engineering and formal methods. The empirical study was conducted in the form of an online survey with anonymous participants out of the described group. Participants were asked to complete the survey without any help, but there is no control mechanism against actual cheating. At the beginning of the survey, we asked the participants for demographic information including their age group, their experience in requirements engineering, HanforPL, and formal logics.

3.3 Object Selection

This first step into the investigation of the understanding of a pattern language is focused on pattern understanding from reading, as it is the basis for further inquiries, e.g., into the generative task of pattern instantiation for formalisation. Therefore, the study (apart from demographic questions) consists of a single repeated task: to decide if pattern instantiations are fulfilled by timing diagrams of system behaviour. Simply checking phrases in isolation (e.g., What is your understanding of the phrase "*holds after at most 2 seconds*"?) was no option, as their interpretation may differ when embedded into the context of a pattern. This can, for example, be seen when comparing the intended meaning of the two phrases of interest (aam) and (aam-cond) within the patterns ***R*** *holds after at most* ***T*** *seconds* and *If* ***R*** *holds, then* ***S*** *holds after at most* ***T*** *seconds.*

Within the survey, we test the participants' understanding of patterns from the HanforPL. To select a suitable set of patterns, the following criteria are considered: 1) The survey should focus on patterns that are relevant in industrial practice, 2) the survey should include the patterns using phrases of interest, and 3) the survey should be short enough to be filled in without too much interruption to a work day of participants in the industry, i.e., the survey should be completed in about 30 to 40 minutes.

We considered patterns that were shown to be used frequently for the formalisation of requirements in the automotive context (criterion 1). We then added patterns containing phrases of interest (criterion 2) if not yet included by the first selection criterion. For patterns whose meaning is inverse to an already added pattern (e. g. *it is always the case that* ***R*** *holds* and *it is never the case*

Question 11-A: Is the requirement "*It is always the case that if **R** holds, then **S** holds after at most **1** second.*" fulfilled in the following example?

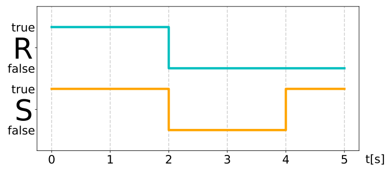


Fig. 2: The first of the four questions to investigate the understanding of the *ResponseDelay* pattern; correct answer: *yes*.

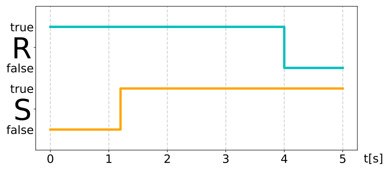
*that **R** holds*), we only included the positive formulated pattern in the survey. We do not assume that negative and positive formulations do behave similar, but that using the usually less legible negative formulation is not adding any new insights. Three patterns adding no unique phrases were dropped due to the timing constraint (criterion 3). The selection process resulted in a list of 17 patterns from the HanforPL (see Table 2).

3.4 Survey Design

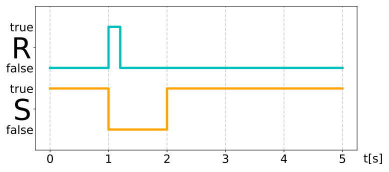
The questions should be formulated in a style that avoids errors based on the incomprehensibility of the survey rather than the pattern under investigation. We therefore decided to work with only one type of question, i. e., we asked whether or not a given instantiated requirement in HanforPL is fulfilled by a given example system behaviour. For each question, the requirement was given as written text, while the example behaviour was depicted as timing diagram. Skipping a question was not permitted. Figure 2 exemplarily shows the first question that was asked to investigate the understanding of the *ResponseDelay* pattern. Consecutively, we asked the same question for three more timing diagrams (Figure 3). That is, for each of the selected patterns, participants of the study had to match four example system behaviours against an instantiated requirement in HanforPL, yielding a total of 68 questions.

The order of questions in the survey and therefore the order of the requirements presented to the participants was static. Participants should be eased into the language by a controlled encounter with the different features of the language, from one observable, over several observables, timed quantification and so on. Thereby preventing noise within the answers resulting from being overwhelmed by a first occurrence of too many new concepts at once. Apart from the gradual exposure to the language features, we assume that no relevant training effect is present, as no feedback on the correctness of the answers was given.

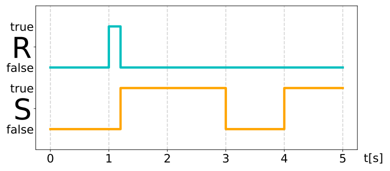
To make the survey feasible within a time frame of about 30 to 40 minutes, the survey includes a high number of example behaviours directly targeting the phrases of interest (see Table 3). Correct answers to these questions thus mean,



(a) Correct answer: *no*.



(b) Correct answer: *yes*.



(c) Correct answer: *yes*.

Fig. 3: Timing diagrams used to investigate the understanding of the *ResponseDelay* pattern (Questions 11 B - D).

that the general behaviour of the pattern has been understood *and* the phrase of interest was interpreted correctly (with respect to the formal semantics).

The survey does not investigate the understanding of different scopes. This would introduce another level of complexity and hence require more questions to be asked to infer reasons for possible incorrect answers. We therefore implicitly instantiated all requirements with the scope *globally*.

4 Results

The study was completed by 37 participants with an average experience in requirements engineering of 3.3, in HanforPL of 1.8, and in formal logics of 3.9 on a self assessment scale of 1 (not experienced at all) to 5 (very experienced). The median age group was *41 to 50*. One participant indicated that they clearly misunderstood the given task as part of a feedback email. The described answer set (all *false*) was clearly identifiable, and the participant was removed as an outlier. Table 2 shows the detailed performance of all participants over all patterns and questions.

Participants had to rate their familiarity with HanforPL in the beginning of the survey (see Figure 4). To investigate Research Question R1, we separate the participants into two groups: The 26 participants being untrained in HanforPL (answering 1 in the related self assessment question) answered with 75% accuracy (on average 51.1 of 68 questions answered correctly). The 10 participants that received former training in HanforPL (answering > 1 in the related self assessment question) answered with 79% accuracy (on average 53.7 of 68 questions answered correctly).

Table 2: Survey results per pattern (listed in the order they occur in the survey) and question (columns A,B,C,D).

	Average of correct answers (%)				
Pattern Name	A	B	C	D	Total
Universality	94	100	100	100	99
Invariance	67	97	64	89	79
Initialization	94	100	100	83	94
Persistence	75	100	86	100	90
Precedence	97	53	78	89	79
DurationBoundL	14	100	92	81	72
DurationBoundU	89	14	92	97	73
ReccurrenceBoundL	97	83	86	92	90
UniversalityDelay	53	92	53	86	71
InvarianceBoundL2	64	58	92	61	69
ResponseDelay	47	89	81	89	76
ResponseDelayBoundL1	58	75	92	53	69
ResponseBoundL1	39	94	53	53	60
ResponseBoundL12	50	100	92	83	81
EdgeResponseBoundL2	97	56	17	86	64
EdgeResponseBoundU1	42	72	86	11	53
EdgeResponseDelayBoundL2	100	78	75	56	77

There is a slight, non-significant trend of training in HanforPL leading to more correct answers (Pearson correlation of $r(34) = 0.292$ with $p = 0.083$). The difference between both groups is statistically not significant (Mann-Whitney-U $U = 103$ with $p = 0.348$). As both groups performed similar, we do not discern between them in the following.

In the beginning of the study, participants had to give a self assessment of their experience in formal logics as well as requirements engineering (relating to R2). We assume that both disciplines give a solid foundation (be it in vocabulary or concepts) for a better understanding of requirements pattern languages.

It turned out, that training in requirements engineering does at best show a weak and statistically not significant trend (Pearson correlation of $r(34) = 0.231$ with $p = 0.175$). Astonishingly, the best and worst participants claimed to have a high understanding for requirements engineering (see Figure 5).

In contrast, experience in formal logic turned out to have a strong correlation (Pearson correlation of $r(34) = 0.647$ with $p < 0.0001$) with the number of right answers (see Figure 6).

As the final research question (R3), we investigate the phrases of interest. Detailed results from the relevant questions can be seen in Table 3. For each phrase of interest, its related patterns and questions, the table shows the overall result, as well as the results of participants with prior training in formal logic (answering > 2 in the related self assessment question; $n = 30$) and with little to no training in formal logic (answering ≤ 2; $n = 6$).

Table 3: Correctness results for the phrases of interest. Each row shows the according phrase id, the pattern containing the phrase and which question in the survey prompted that exact behaviour followed by the percentage of correct answers. Column N shows participants with little to no, column L with training in formal logics.

			Correct		
ID	Related pattern	Question	N (6)	L (30)	Overall
prev	Precedence	C	50	83	78
prev	Precedence	D	67	93	89
afterw	ResponseBoundL1	D	0	63	53
afterw	EdgeResponseBoundU1	B	50	77	72
afterw*	ResponseBoundL1	C	33	57	53
afterw*	ResponseBoundL12	A	33	53	50
afterw*	EdgeResponseBoundU1	A	50	40	42
aam	UniversalityDelay	A	33	57	53
aam	UniversalityDelay	C	33	57	53
aam-cond	ResponseDelay	D	83	90	89
aam-cond	ResponseDelayBoundL1	C	83	93	92
obs	DurationBoundL	A	17	13	14
obs	DurationBoundU	B	0	17	14
obs	EdgeResponseBoundL2	C	0	20	17
obs	EdgeResponseBoundU1	D	0	13	11
obs+	DurationBoundL	C	100	90	92
obs+	DurationBoundU	D	100	97	97
obs+	EdgeResponseBoundL2	D	83	87	86
obs+	EdgeResponseDelayBoundL2	B	83	77	78
rec	ReccurrenceBoundL	B	83	83	83

Table 4: Remainder of questions with high error rates not already covered by the phrases of interest. Column N shows participants with little to no, column L with training in formal logics.

			Correct		
ID	Related pattern	Question	N (6)	L (30)	Overall
antec	Invariance	C	17	73	64
antec	InvarianceBoundL2	D	17	70	61
atonce	Precedence	B	50	53	53
atonce	ResponseDelay	A	33	50	47
atonce	ResponseDelayBoundL1	A	33	63	58
atonce	ResponseDelayBoundL1	D	50	53	53
atonce	ResponseBoundL12	A	33	53	50

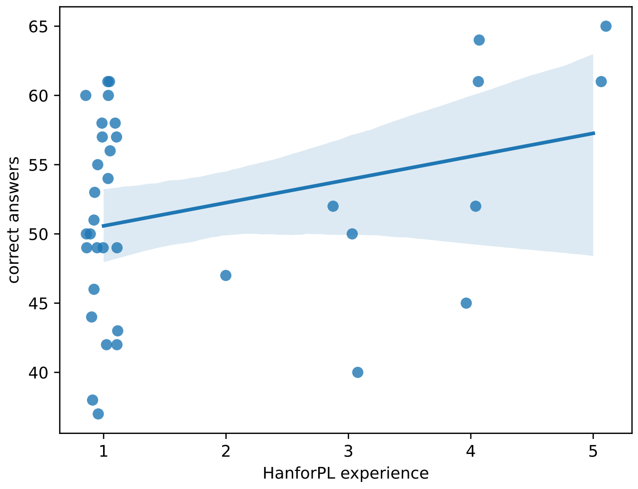


Fig. 4: The influence of former training in HanforPL (x-Axis) on the number of correct answers given (y-Axis).

Table 4 contains results of the remainder of questions with high error rates. The errors from these questions can be attributed to two kinds of formulations and underlying semantics used in the pattern language. For ease of reading, we define these ad-hoc categories analogous to the phases of interest:

(antec) *[...] if **R** holds, then **S** holds as well* : This requirement's semantic is equal to the implication $R \rightarrow S$, i.e., if ***R*** has to hold, then ***S*** has to hold as well, but not vice versa.

(atonce) *[...] if **R** holds, then **S** holds after at most **T** time units*: In this example, it is not clear if **S** is expected to be in real succession to **R** (as one would expect for a causal relationship), or if both happening at the same time is also valid behaviour. The latter is the case in HanforPL.

5 Discussion

The overall results regarding the understanding are positive, showing that most patterns in HanforPL can be understood even without prior training in the pattern language.

Results of 75% to 79% correct answers of participants untrained and trained in the pattern language entail that generally more than every fifth answer to questions of whether behaviour belongs to the system are erroneous. This interpretation is heavily skewed as the survey is focused on phrases of interest, i.e., on edge cases which are prone to misinterpretation. Thus favouring participants familiar with HanforPL as well as skewing the distribution of patterns heavily

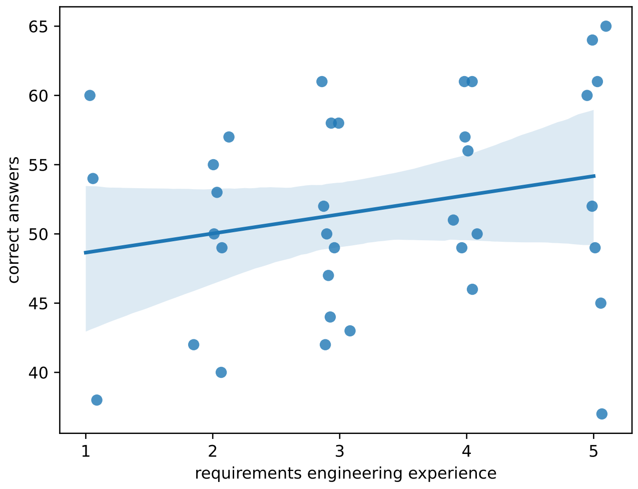


Fig. 5: The influence of experience in requirements engineering (x-Axis) on the number of correct answers given (y-Axis).

towards more complex patterns within the survey, that are used with far lower frequency in practice. Requirements sets usual for industrial practice, as reported by [12], mainly contain patterns that got high success rates. This is especially the case for the *Universality* pattern and common applications of *Invariance-BoundL2* and *ResponseDelay*, i.e., excluding the answers to question A of the latter pattern, (see Table 2). Therefore, we conclude, that HanforPL turned out to be understandable, even for untrained participants.

Results show, that training in formal logic serves as a good predictor for the comprehension of the requirements pattern. The explanation of this effect could be twofold: For one, formal logics, especially temporal logics (e.g. LTL, MTL or Duration Calculus) have similar interpretation of concepts like referring to, e.g., a future state just requires just a non-zero interval (or one state) except denoted differently. Thus, the everyday understanding of these terms is already aligned with the formal meaning. Second, training in formal logics (in contrast to requirements engineering) may allow for more detachment from the actual physical system, i.e., ignoring the question as to what might happen before or after the timing diagram.

Analysis of individual phrases allows to pinpoint phrases and concepts that are not aligned with their everyday understanding (Table 3).

The results show, that (rec) and (prev) are unproblematic, as questions regarding those phrases of interest were answered correctly by most participants.

For the phrase of interest in (afterw), i.e., the text ***S*** *holds afterwards*, participants leaned on the side of ***S*** only holding for a non-zero interval which matches the intended meaning (with 53% resp. 72% correct answers). For the

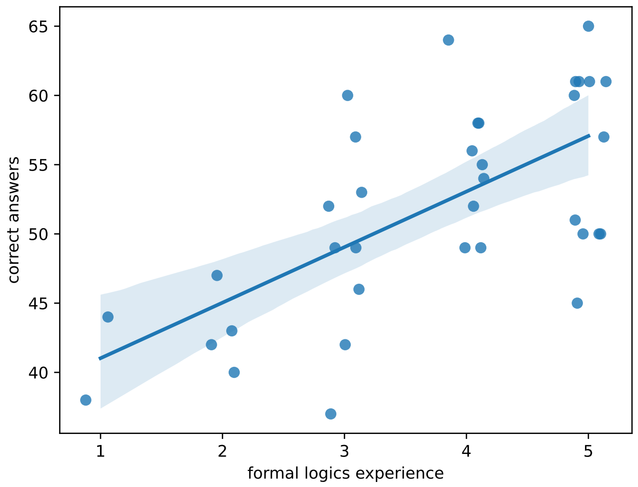


Fig. 6: The influence of experience in formal logics (x-Axis) on the number of correct answers given (y-Axis).

ResponseBoundL1 D question, the divide between logically trained (63% correct) versus untrained (0% correct) shows that there is a different understanding of the phrases depending on training, i.e., all of the latter did assume that ***S*** has to hold persistently. Again, disambiguation by including the word *persistently* in pattern where this is the case should solve this case.

Regarding (afterw*), the question whether ***S*** has to hold immediately after the trigger event (intended meaning), participants leaned to answer incorrectly (with 53%, 50%, resp. 42% correct answers). This result shows, that the behaviour has to be made explicit. The uncertainty if ***S*** has to hold immediately or at some arbitrary point (afterw*) should be addressed by including the word *immediately* as part of the patterns.

All participants performed well on the phrasing of (aam-cond) *if [...], then **S** holds after at most **T** seconds.* In contrast, for (aam) only 53% answered correctly, i.e., that the observable has to hold persistently. Thus, the interpretation in (aam-cond) is in alignment with the common understanding, while the *UnversalityDelay* pattern containing the (aam) phrase should be changed to include the phrase *persistently* to be *[...] **S** holds after at most **T** seconds persistently.*

The most recent addition to the pattern language is concerned with reaction to changes of observables. Questions related to the phrase *once **R** becomes satisfied, [...]* (obs+) were consistently answered correctly, i.e., the requirement has to be evaluated after each time **R** becomes satisfied.

The question if an explicit rising edge is required (obs) and how especially initial behaviour is treated was highly problematic (below 17% correct answers). Answers were systematically given so, that the state of the system before the timing diagram was the missing part to satisfy the change of the observable. As we did not alter the observables, we did not include the negative case. Including the negative case would have been beneficial in analysing if participants just assumed that all observables are *false* in the beginning, or if any state was possible that suited the interpretation.

The detailed results in Table 2 show a number of questions that turned out to have a high error rate. We assigned additional ad-hoc phrases of interest: Low rates of right answers in (antec) (see Table 4) could be attributed to a common error when dealing with implications, the *denying the antecedent*. For example, the pattern *it is always the case that if* R *holds, then* ***S*** *holds as well* is satisfied by the behaviour depicted in Figure 7. Nonetheless, ***S*** being true without ***R*** being true in time interval $[2, 3]$ was seen as a violation by 36% of participants, especially those with little to no training in formal logic (only 17% correct in both questions). For nine participants (25%) the error was stable over both questions regarding (antec). This could point to a systematic misunderstanding of implication, or at least a difference in the understanding to the phrasing used for implication in this pattern. The existence of systematic differences of understanding conditionals has been shown by Fischbach et al. [6].

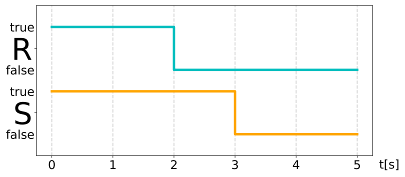


Fig. 7: Example of a *denying the antecedent* error in the survey.

A large number of errors stem from cases in which everything relevant happens at the same point in time (atonce). An example is the requirement *if* ***R*** *holds, then* ***S*** *holds after at most* ***1*** *second* together with the behaviour depicted in Figure 8. One can see that for time interval $[0, 2]$ ***R*** as well as ***S*** are true, i.e., the causal relation, although it only needs to be satisfied with a delay of at most one second, is satisfied immediately. This may be again due to a notion of the requirements as more of a physical system, where the trigger results in an action with a real causal delay.

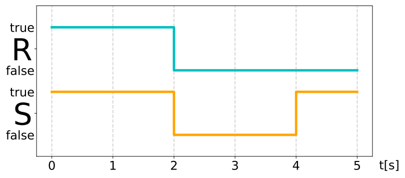


Fig. 8: Problems with immediate satisfaction of a property.

Many of the problems detected in this study should be fixed by small changes regarding the pattern language. As an immediate result of this study, several improvements for the phrases of interest were suggested, as discussed above. These modifications have to be verified carefully so, that the simplicity of the sentences is not lost in an overly complex sequence of adjectives describing each observable.

Similar to the argument in [3,14], a basic understanding of formal logic (better formal methods in general) should be the best mitigation for misalignment in the understanding of formal constructs such as the problems found with (antec).

Additionally, we include clarifications targeted on the misunderstandings found in this survey in our training material.

6 Threats to Validity

6.1 Internal Validity

The threat of *Repeated Testing* is concerned with participants learning over the run of an experiment. As participants were not informed if their answers were correct, they should not have been able to gain information on the correct interpretation of the pattern. An acclimatisation to the pattern language was intended though in order to prevent participants from being overwhelmed with the more complex pattern. As the survey was performed in an industrial context, *Maturation*, i.e., changes over the duration of the survey can influence the results. We tried to keep the survey as short as possible in order to prevent tiring and impatience (or loss of participants) due to more pressing concerns. The threat of *Instrumentation* is concerned with the influence of the experimental material itself on the results. We tried to make the examples of system behaviour as accessible as possible for the use by not formally trained participants [4]. Nonetheless, problems with phrases of interest starting in the beginning of the timing diagram may have suffered from a notion of the system too commonly associated with a real system, i.e., where there is always a previous state even if switched off. To guarantee that the questions themselves do not contain errors, all timing diagrams were automatically verified by the pattern simulator being part of Hanfor.

6.2 Construct Validity

The threat of *Interaction of setting and treatment* is concerned with non-aligning circumstances of experiment and reality. In fact, the experiment is presented in a form focusing on the patterns itself, not on realistic requirements. In a real setting, expressions over observables in pattern instantiations can add another layer of complexity, that is abstracted away, to get data on the pattern themselves. In reality, the correctness numbers can be much lower as requirements get considerably more complex. Nonetheless, the expression language is not likely to have interactions with the phrasing of the surrounding pattern.

6.3 External Validity

The threat of *Interaction of selection and treatment* is concerned with the selection of non-representative participants. Our participants were selected by contacting cooperation partners from different engineering divisions, and the chair mailing list. This way, we tried to spread the risk of convenience sampling over different businesses and person groups likely to be in a position or likely to be in the near future of using a requirements pattern language.

7 Related Work

Winter et al. [15] conduct a survey on the understandability of quantifiers and their negation (such as *all*, *more than* or *at least*) in natural language requirements. Results show, that there are significant effects on reading speed and error rate between the different quantifiers and their negated forms. Based on the results, advice for writing requirements is given. This recent work shows the relevance of investigations into the understanding of requirements in general. Phrasings are chosen once and reproduced in each instantiation, i.e., any problem introduced to a pattern is multiplied over a requirements specification. Therefore, ensuring understanding by a broad audience is even more relevant.

Giannakopoulou et al. [7] address the problem of pattern understanding by presenting several representations of the instantiated requirement, both graphical and as formal logic, e.g. LTL. This is a necessary support for error recovery by comparison to the intended result, while the pattern language should itself prevent errors in the first place by being aligned with the intuitive understanding of the patterns.

A different approach is taken by Moitra et al. [10], designing the requirements language in the style of a programming language. This surely aligns the intuitive understanding with stakeholders from a computer science background, but may exclude other stakeholders entirely, because of the condensed syntax.

8 Conclusion

In this paper, we demonstrated how an inquiry on the alignment of the formal semantics of the HanforPL and the intuitive understanding of requirements engineers can help to understand and improve the pattern language. Almost half of the patterns considered in the survey are contained in the SPS by [13]. Parts of the results can therefore be generalized to SPS-like languages.

The analysis results are positive, and the pattern language performed very well in hiding the formal complexity behind intuitively understandable sentences. Nonetheless, the language contains several phrases that lead to near random decisions, and misconceptions of logic can lead to misinterpretations that cannot be mitigated entirely by phrasing. We suggested several improvements through the analysis.

This study was a short, industry friendly foray into the comprehensibility of HanforPL. In the short term, the pattern of HanforPL will be improved by the suggested changes. Based on this study, future work will be to design a more thorough investigation of the patterns, especially in conjunction with scopes. The basis of this extended survey could be a mutation based scenario generator to do the tedious work of generating different classes of scenarios. While the examination of each pattern is of immediate use for requirements engineering, the question remains if we can evaluate the meaning of single words (e.g. *after* and *once*), or if their meaning is heavily influenced by the context in which they occur.

Acknowledgements

We thank all participants, and Amalinda Post for forwarding the study.

References

1. Becker, S., Dietsch, D., Hauff, N., Henkel, E., Langenfeld, V., Podelski, A., Westphal, B.: Hanfor: Semantic requirements review at scale. In: REFSQ Workshops. CEUR Workshop Proceedings, vol. 2857. CEUR-WS.org (2021)
2. Berry, D.M., Kamsties, E.: The syntactically dangerous all and plural in specifications. IEEE Softw. **22**(1), 55–57 (2005)
3. Bjørner, D., Havelund, K.: 40 years of formal methods - some obstacles and some possibilities? In: FM. Lecture Notes in Computer Science, vol. 8442, pp. 42–61. Springer (2014)
4. Dietsch, D., Feo-Arenis, S., Westphal, B., Podelski, A.: Disambiguation of industrial standards through formalization and graphical languages. In: RE. pp. 265–270. IEEE Computer Society (2011)
5. Dietsch, D., Langenfeld, V., Westphal, B.: Formal requirements in an informal world. In: 2020 IEEE Workshop on Formal Requirements (FORMREQ). pp. 14–20. IEEE (2020)
6. Fischbach, J., Frattini, J., Mendez, D., Unterkalmsteiner, M., Femmer, H., Vogelsang, A.: How do practitioners interpret conditionals in requirements? In: PROFES. Lecture Notes in Computer Science, vol. 13126, pp. 85–102. Springer (2021)
7. Giannakopoulou, D., Pressburger, T., Mavridou, A., Schumann, J.: Generation of formal requirements from structured natural language. In: REFSQ. Lecture Notes in Computer Science, vol. 12045, pp. 19–35. Springer (2020)
8. Konrad, S., Cheng, B.H.C.: Real-time specification patterns. In: ICSE. pp. 372–381. ACM (2005)
9. Langenfeld, V., Dietsch, D., Westphal, B., Hoenicke, J., Post, A.: Scalable analysis of real-time requirements. In: RE. pp. 234–244. IEEE (2019)
10. Moitra, A., Siu, K., Crapo, A.W., et al.: Towards development of complete and conflict-free requirements. In: RE. pp. 286–296. pubIEEE (2018)
11. Post, A., Hoenicke, J.: Formalization and analysis of real-time requirements: A feasibility study at BOSCH. In: VSTTE. Lecture Notes in Computer Science, vol. 7152, pp. 225–240. Springer (2012)
12. Post, A., Menzel, I., Hoenicke, J., Podelski, A.: Automotive behavioral requirements expressed in a specification pattern system: a case study at BOSCH. Requir. Eng. **17**(1), 19–33 (2012)
13. Post, A.C.: Effective correctness criteria for real-time requirements. Ph.D. thesis, University of Freiburg (2012)
14. Westphal, B.: On education and training in formal methods for industrial critical systems. In: FMICS. Lecture Notes in Computer Science, vol. 12863, pp. 85–103. Springer (2021)
15. Winter, K., Femmer, H., Vogelsang, A.: How do quantifiers affect the quality of requirements? In: REFSQ. Lecture Notes in Computer Science, vol. 12045, pp. 3–18. Springer (2020)
16. Yang, H., Roeck, A.N.D., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requir. Eng. **16**(3), 163–189 (2011)