Received: 30 July 2020 | Revised: 29 March 2021 | Accepted: 26 April 2021 | IET Cyber-Physical Systems: Theory & Applications

DOI: 10.1049/cps2.12017

ORIGINAL RESEARCH PAPER

Table-based formal specification approaches for control engineers—empirical studies of usability

Suhyun Cha[1] | Birgit Vogel-Heuser[1] | Alexander Weigl[2] | Mattias Ulbrich[2] | Bernhard Beckert[2]

[1]Chair of Automation and Information Systems, Technical University of Munich, Garching, Germany

[2]Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany

Correspondence

Suhyun Cha, Chair of Automation and Information Systems, Technical University of Munich, Boltzmannstr. 15, 85748, Garching, Germany.
Email: suhyun.cha@tum.de

Funding information

Deutsche Forschungsgemeinschaft, Grant/Award Number: VO 937/28-2, BE2334/7-2, UL 433/1-2

Abstract

The dependability characteristic of the control software of manufacturing systems is highlighted more than before, going through repeated changes to cope with various and varying requirements. Formal methods are researched to be applied to automation system engineering to obtain a more effective and efficient quality assurance. One of the approaches, a formal specification language named Generalised Test Tables has been developed with the aim of intuitiveness and accessibility for automation application developers. The result of the experiments conducted to assess the usability of this language is presented here. Focussing on evaluating effectiveness and user satisfaction, three paper-based experiments have been conducted with students at the bachelor and master level. The evaluation results point to positive usability in both comparative effectiveness to conventional language, that is, Petri Nets, and subjective perception of user satisfaction.

1 | INTRODUCTION

Cyber-physical production systems (CPPS), including manufacturing machines and logistics, have become more complicated following current trends such as increasing customer flavour variety [1]. CPPS, also called as automated Production Systems (aPS) [1], are Cyber-Physical Systems applied in the domain of manufacturing or production [2]. The proportion of system functionalities realised by the software in aPS is increasing [3] to benefit from its software's flexibility, among various disciplines, namely mechanics, electrics/electronics, and software [4]. Malfunctioning control software of such systems may cause damage to the system itself, to payloads (and correspondingly customers), or even to persons within the reach of the system; therefore, sufficient software quality assurance is essential [5]. aPS are usually automated with Programmable Logic Controllers (PLCs), and these computing devices with control software are expected to control aPS with assured quality in dependable and safety–critical real-time environments [6].

In today's industrial practice, control software quality is commonly achieved by dynamic validation either through step-by-step manual testing or automatically generated test cases [7]. However, the main weakness of traditional testing is that one test case covers only a single, particular run of the aPS software. This implies that system behaviour is often not fully explored during validation and that some scenarios remain untested, which might lead to an incomplete proof of software correctness. Unpredictable and rare malfunctions may therefore remain, and this can have severe consequences. In contrast to testing, formal verification achieves full coverage by proving the correctness of an implemented programme mathematically and exhaustively regarding the given specifications. Moreover, formal verification in comparison could be suitable to satisfy the requirement of guidelines or regulations related to reliability, utilising its completeness in the process of re-validation, which is often required by regulations for each change of a safety–critical aPS [8]. Thus, there is a need to support formal verification of PLC software [9, 10].

Yet, formal verification is not commonly used to verify the correctness of a control software [11–13]. One of the main reasons for this is that it is difficult to specify the desired temporal properties of a system since that requires expert knowledge of high mathematical training in formal specification languages [14, 15]. Even worse, in many cases, an informal description of the requirements that could be used as a basis for a formal specification [8] is not even available. The use of a well-defined specification language could provide the additional benefits of maintenance, reuse, and trouble-shooting as well as formal verification [11]. Ljungkrantz et al. [11] even claim that 'today's need for control software developers to understand mechanical and electrical drawings might also be less important if better specifications were available.'

To tackle this problem, we have suggested a formal specification language in the recent works [16, 17], namely Generalised Test Tables (GTTs). The origin of GTT lies in test tables that are the industry's practices to describe test cases in the table form [18, 19]; and GTT can be used as a formal language for verification purposes supplementing testing by generalising test tables. The technical applicability of the language has been demonstrated and evaluated in [8] regarding formally analysed expressiveness and as a result of verification experiments using a set of demonstrator scenarios.

A formal specification language works as a communication medium between peers [20] of the requirement, the verifier, and the user of the verification. Therefore, usability is as important as the functional effectiveness of the language. While these types of languages that provide concepts and notations tailored for a specific purpose [21] are being developed to obtain higher productivity in system engineering through appropriate notations and abstractions, the usage rate of these is comparably low often due to the frequent absence of usability validation [22, 23]. Thus, the need to understand the usability of the developed language leads to evaluation initiatives valuing the result by following the standard definition of usability [24]. The International Organisation for Standardisation (ISO) [25] defines usability as 'the extent to which a system, product, or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.' (Note: This is a definition from ISO/IEC 9241-210. Standards including usability, e.g. ISO 25010:2011—also known as withdrawn ISO 9126-1, refer to this definition.) With regard to these three factors, we focussed on effectiveness and user satisfaction.

The evaluation experiments presented in this paper were conducted to understand effectiveness and user satisfaction. Effectiveness means the possibility of accomplishing a successful task and that a user should be able to comprehend and create artefacts in the (modelling) language to an acceptable degree, as summarised by Schalles et al. in [26] in this regard. User satisfaction means the degree to which user needs are satisfied in the usage [27] (it is defined similarly in different works of literature such as [26, 28]). In three experiments, potential users of the developed language participated in the training, exercises, and a user perception questionnaire for the analysis of (i) how much effectiveness is achieved compared with conventional language, (ii) how much satisfaction the users perceive during usage of the language; and (iii) assessed feedback to elaborate the developed language coming up with the users' perspective.

The paper's outline is given as follows: First, the approaches of domain-specific languages for formal specifications are reviewed together with usability evaluations in Section 2. In Section 3, the target language, GTTs, is introduced. In Section 4, research questions (RQs) and corresponding hypotheses are stated for the experiments. The experiment plans and results are presented in Section 5 and 6, respectively, followed by the summary of the hypotheses' proofs and a discussion of validity in Section 7. Finally, the paper closes with the conclusion and an outlook in Section 8.

2 | RELATED WORKS

Since this paper presents the evaluation of a developed formal specification language's usability, this section addresses states of related research works about formal specification approaches of automation system in Section 2.1 and their usability studies in Section 2.1.

2.1 | Formal languages used in formal verification of aPS control software

Although formal verification of aPS behaviour is a comparably recent research topic, some works have focussed on verifying PLC software using model checkers with conventional logics as specification from [29] up to [30]. Some research already targeted formal verification of the IEC 61131-3 programme (e.g. [31–33]). Specification relaxation is another stream to ease the specification elaboration to benefit from counterexample, for example, [34] as a recent approach.

Formal languages to specify desired behaviour have also interested researchers. Dwyer et al. [35] present initial work applicable for industrial cases as property specification patterns. Pattern-based approaches have been tailored for automation systems with regard to safety patterns for production systems engineering (e.g. [36]) or for automotive control units (e.g. [37]). More accessible to engineers, graphical specification languages are developed and used. Grafcet [38] is a widely used standard due to its accessibility and still researched for modelling (e.g. [39]). Timing diagrams have also been considered as a major graphical language as utilised in [7] or [40] exemplarily. Darvas et al. [41] presented a formal specification language for PLC programs aiming at code generation, although it lacks details of state change descriptions. Przigoda et al. [42] utilised Unified Modelling Language (UML) and Object Constraint Language (OCL) models to be used for property verification. GTTs [16, 17] are suggested recently as a formal specification language of reactive systems, covering usability limitations. This approach is motivated by test tables suggested by industry, as they are preferable and introduced in Section 3.

2.2 | Usability study of specification languages

Domain-specific languages provide concepts, notations elaborated for a specific purpose, and higher productivity in system engineering [21]. Since model-based engineering methodologies are pervasive, metamodels and languages appear in various forms to satisfy goals and needs. Focussing on specific aims often hinders the actual usage aspects and then the language is only left in theory. There are studies regarding the usability of modelling languages that are usually used to design systems or the usability of a tool itself regarding the user interfaces (e.g. [43]). Since the GTT approach aims at a formal specification, some works focussing on the evaluation of specification languages are stated here.

Teruel et al. [44] present a brief result of usability evaluation for a requirement modelling language showing a completeness rate and a user satisfaction analysis. Snook and Harrison [45] show the similar comprehensibility between a formal specification language and Java. Carew et al. [46] experiment for the acceptance level of formal and informal specification and training time for formal language. Razali et al. [47] present the usability experiment result of the combination of semi-formal and formal language with regard to comprehensibility and preferences, and conclude that the combination is useful in promoting specification. Timing-diagram-based language implies usability of the conventional timing-diagram, and pattern-based approaches (e.g. [36, 48]) are also regarded as usable, initiated by categorising existing specifications and extracting the patterns. Focussing on this aspect, Pakonen et al. [15] review the accessibility of several specification languages regarding coverage of predefined formal properties as effectiveness evaluation in terms of ISO standards. Empirical evaluation result is reported including the subjective confidence aspect, for a diagrammatic representation in [49]. Another modelling language is presented together with a brief notation evaluation in [50]. Unfortunately, not many usability evaluation studies could be found for most of the specification languages stated in the previous section. Even though the developed or extended language is regarded to entail a certain level of usability implicitly inherited from the origin, the usability of the developed language should be re-evaluated for its formation as well as objective changes.

3 | TARGET LANGUAGE

A GTT is a formal specification language designed to represent a set of allowed reactive systems' behaviours, originated from concrete test tables that are test cases from industry practices. A test case indicates a sequence of concrete input and output value pairs $(I \times O)$ where I and O describing the domain of input and output values of the system are under test. Each entry (i,o) in $(I \times O)$ is a step in the test where i is the concrete input stimuli to the system, and o is the expected response. Since the output of the reactive system depends not only on the current input, but also on the input history, a description in a table form is used to specify steps (rows) in which values are passed to inputs, and the resulting output variables of the system (columns).

While a test table represents a single test case with concrete values in the cells, GTTs describe a family of test cases by substituting concrete values with constraint expressions based on generalisation concepts (cf. Figure 1: Full details of the language, including the syntactic and semantic formalisms, can be found in [8, 17])

- Abstraction: cells may contain constraint expressions to represent possible concrete values as described 'value in the interval from 0 to p' in [#3:X], or 'value greater than Y/2' in [#3:Z].
- Referencing: corresponding relations among cells can be expressed using global variables [#2:B], for example, [#2:C], [#2:X], [#3:B], and [#3:X] with value constraints related to 'p', or cell indices (i.e., column names) used in other cells, for example [#2:Y], [#2:Z], [#3:Y], and [#3:Z]. Referring by column name can be expressed with designating relative cycles such as '[-n]' for the n-th previous scan cycle.
- Duration generalisation: possible repetition cycles and rows are specified using a duration constraint describing lower and upper bound of the execution, for example [#2:Duration], meaning that the row must be repeated more than five times.
- Block repetition: some consecutive rows can be grouped with a specific symbol with its own duration (or repetition) constraint. Row groups can be nested.

Formally, a GTT represents a regular expression over the pairs of pre- and post-conditions $(\varphi, \psi) \in C_\Sigma$ where C_Σ is the set of all constraints over variable signature Σ consisting of the input and output variables of the system. A tuple (φ, ψ) models a single row, where the constraints for the input (φ) and output (ψ) are conjunctively joined. The row and group repetition is expressed via the repetition operators. Hence, a GTT describes the set S of sequences of pre-and post-conditions, that is, $S \subseteq (\varphi, \psi)^*$. Considering all possible value combinations of constraint in G, the conformance of a system against a GTT G is defined as conforming to Σ. Until the end of the table or invalid input (regarding the pre-conditions) is received, the system conforms with an instantiated GΣ if and only if its outputs do not violate the sequence of post-condition. This concept has been defined with a two-party game between a reactive system and a GTT in [17].

Compared to the other approaches indicated in Section 2 exemplarily, this GTT approach has been motivated by the industry practices, that is, test tables, which are also embedded within one of the most common IEC61131-3 development environments (CODESYS Test Manager [51]). Although various formats have been suggested as intuitive methods, many have been rejected to be used in day-to-day aPS engineering activities mainly due to its additional effort for the automation engineers to get trained. In this context, this paper targets to investigate how much the developed approach accords with the main objective; and this will be discussed in detail in the next section.

4 | RESEARCH QUESTIONS AND HYPOTHESES

A newly developed language would be promoted more for its usage in engineering activities if it were evaluated with regard to its usability as well as its technical utility and if it were evolved based on evaluation analyses. To accomplish such

#	Input			Output			Duration
	A	B	C	X	Y	Z	
1	1	1	FALSE	0	0	-	1
2	-	p, > 1	FALSE	[0, p]	X	Z[-1]	>5 [1,3]
3	-	p	TRUE	2*p	>Y[-1]	>Y/2	-

FIGURE 1 An example of a Generalised Test Tables (GTT; derived from [16])

purposes, we have formulated the following RQs regarding usability of GTTs under consideration of effectiveness and user satisfaction:

- RQ1: Is the target language effective for the purpose?
- RQ2: Is the target language satisfactory for automation engineers?

Considering major usage activities of the specification language, as applied in [52] and described in [53], effectiveness could be defined as the ability to understand, create and learn perspectives. Looking at each of these chronologically, creating is important to represent the intention of the requirement (properties) in the viewpoints of the automation engineers, that is, developers or testers of the automation system software, as grounds for verification. Also, understandability is important for developers to figure out whether the existing specifications depict the property correctly or how the specifications diverge from the intention. For the proper use of the language, the user must understand it correctly with regard to its syntaxes and semantics.

To measure usability with regard to these aspects, an evaluation of the control language, Petri Nets (PN), was also conducted to see the comparable dominance compared with the conventional language objectively instead of solely measuring the absolute degree. For the tasks, the language should be able to deliver the originally intended semantics to the user through the understanding activity. This can be assessed by observing whether users comprehend the intention of the specified descriptions in the language (H1.1). In reverse, the language should be able to appropriately describe the intended behaviour by the user. This can be assessed by observing whether users use defined notations of the language properly to describe the given behaviour (H1.2). If the understanding and creation of activities regarding the target language depend on the specific knowledge or experience of the user, the aim of user accessibility is faced since the successful result of these activities would be affected by individual knowledge or background levels, not by the language itself, and the domain experts with prior knowledge are not necessarily the language's end users [20] (H1.3). Additionally, scalability of the language is to be considered with the assumption of usage for practical cases. Although it would be systematically possible to describe enlarged behaviour, comprehension and creation of the scaled version by users, is assessed separately (H1.4).

The degree of user satisfaction is another aspect to be researched with regard to the usability of the language. Based on the system usability scale items [54] in consideration of the major activities of language usage, users must have confidence regarding comprehension and creation after some experience in using it (H2.1, H2.2). From the same point of view, the degree of experience for learning should not be regarded as very large (H2.3). In addition, the degree of satisfaction would be related to the users' will to utilise it as a solution or, more specifically, as a tool to achieve the solution (H2.4, H2.5). The hypotheses regarding each RQ are summarised in Table 1.

5 | EVALUATION EXPERIMENT

Over the 2019 summer semester and the 2019–2020 winter semester, three experiments were conducted with master and bachelor students of the mechanical engineering department at the Technical University of Munich. The experiments were conducted in a paper-based manner since the application was still in development. The subsections describe each of the experiment designs and participants. All the programme codes given to students during the experiments were in IEC 61131-3 Structured Test.

For the control language in the comparison experiment, Condition/Event (C/E) type PN [55] was selected among various types of PN for two reasons. First, C/E PN has a similar mechanism to represent the behaviour evolution of the system without handling tokens to avoid unnecessary disadvantages in the control language. It consists of the conditions and resulting events without tokens. The conditions could be comparable with the input section of the GTT and the resulting event to the output section. As long as the GTT does not have a divergence, **it** works similar to the GTT since one behaviour path is indicated as GTT. Second, the participants were well aware of it since it had been taught in another mandatory course as a system behaviour description language. To cover the participants who did not know about it, a corresponding lecture had been conducted before the experiment (cf. Section 5.2).

The given tasks for the evaluations are developed based on a lab-sized manufacturing plant demonstrator, extended Pick & Place unit (xPPU) [56] which has been invented to benchmark evolution scenarios of aPS. Although xPPU is quite simple comparable with the actual plant system, it realises the basic functionalities representative as identified by Spindler et al. [57]. Tasks in the evaluation are based on some of the scenarios, component parts, or comparable machines from the actual machines.

5.1 | Comparing the effectiveness of GTT and PN concerning understanding and creating with separated participant groups (Experiment 1)

This experiment (Exp-1) consists of a one-hour tutorial of the target language, that is, GTTs, given to the participants in the form of a lecture allowing them to ask questions and including a review of examples. Tasks were given to the participants

TABLE 1 Research questions (RQs) and related hypotheses

RQ	Related hypothesis	Proof method
RQ1: Is the target language effective for the purpose?	H1.1: A user can **understand** a system behaviour by the specifications in GTTs similar to or better than by the conventional language (Petri Nets [PN]).	Score comparison of the understanding specification task in GTTs and PN
	H1.2: A user can **create** a specification regarding a system behaviour in Generalised Test Tables (GTTs) similar to or better than in the conventional language (PN).	Score comparison of the creating specification task in GTTs and PN
	H1.3: **Understanding** and **creating** specifications in GTTs are less related to the **background knowledge** of the user regarding software engineering than to the conventional language (PN).	Correlation study between the understanding task score and the personal grade
	H1.4: The score of the **understanding** task is less sensitive to the **scale** of the specification in the target language than the conventional language (PN).	Correlation study between the specification complexity and the score comparison of the understanding task
RQ2: Is the target language satisfactory for automation engineers?	H2.1: Users have confidence in **understanding** a specification in GTTs.	Questionnaire
	H2.2: Users have confidence in **creating** a specification in GTTs.	Questionnaire
	H2.3: Users think the language, that is GTTs, is reasonably **learnable**.	Questionnaire
	H2.4: Users are **satisfied** with the language, that is GTTs as a behaviour specification representation method in general.	Questionnaire
	H2.5: Users would **like to use** the language, that is GTTs, as a behaviour specification representation method in the future.	Questionnaire

separately in the target language (GTT) and the control language (PN) for comparison. The first experiment was also planned as a preliminary measurement of the appropriate level of tasks.

Following paper-based exercises, tasks had to be answered during another hour, one week after the training, due to the timing constraint. To estimate the participants' achievement, two different scales of specifications and target systems were provided as understanding tasks followed by one creating task. The evaluation session was proceeded by the presenter in front of the participants. The participants were divided into two groups and provided with the behaviour described in GTT or PN. Each group received the specification of each language alternately in two understanding tasks, so that they would not be biased by the learning effect of the tasks. A summary of notations of each language was provided to all participants, so that they were not to be biased unnecessarily by not recalling notations.

In task1 (simpler understanding task, 15′), a brief description in the natural language and a formal specification (in GTT for Group1 and in PN for Group2) were provided as information of the target system block. Participants were requested to understand the given information and then find errors in the programme code. Task2 (scaled understanding task, 15′) was framed the same way, but with another system block and the formal specification was assigned the opposite way: in PN for Group1 and in GTT for Group2. The assigned time was set for task1 despite the increased scale since the number of erratic parts was reduced. For task3 (creating task, 30′), a precise description in the natural language and the corresponding programme code were provided as information about the target system block. Participants were requested to understand the given information and then create a formal specification in both GTT and PN. More time was assigned, assuming that creating requires more time than understanding since it requires more processes such as notation diversity, syntax check, and structuring [58].

5.2 | Comparing the effectiveness of GTT and PN concerning understanding and creating with balanced tasks (Experiment 2)

This experiment (Exp-2) was designed based on the analysis of Exp-1 and conducted as part of the class exam with 34 master course students. Learning from the Exp-1 and its result, which will be discussed in detail in the next section, as well as the complexity level between GTTs and the control language (PN), were to be balanced more precisely and correctly. Since the participants had to handle both GTTs and PN as specification languages for each task, the target behaviour of each should not be the same due to the learning effect. In other words, one might recognise behaviour with the easier language for him/herself and try to apply it to the other if the same system is given in two languages. Instead, the system for each language was different, but the complexity of the behaviour described in the language was controlled to be at the same level. For this, each had the same table size when described in GTTs and the same number of places and transitions in PN. It was learnt from Exp-1 that not all the lecture participating students were aware of PN clearly and the knowledge level should be balanced for the comparison, otherwise fewer number of the participants would be regarded as valid which might affect the validity of the experiment results. Thus, a comparable training session for PN was conducted for the lecture students and relevant materials were given as done for GTT.

Two tasks were provided in Exp-2 with subtasks for each. In the first task, a brief description in the natural language and a formal specification in PN were provided. The first sub-tasks are regarding the understanding of the specification consisting of two questions of understanding overall behaviour and tracking some signal pair changes. Different from Exp-1, programme code was not involved in this task on the one hand to simplify the task focussing on the specification, and on the other hand to remove any bias due to programme comprehension abilities. Since the amount of information and questions were reduced, the assigned time was also decreased to 6 min. The second sub-tasks were in regard to creating the specification, which was partially based on a given specification. The creation of repeated behaviour was the special focus in this creation experiment. The partial behaviour was thus indicated as a repeated part on the given specification and participants were asked to generate the change on the sheet. In PN, two options were given: to implement the change with a counting variable (easy) and by adding places and corresponding edges (hard) to see at which level the GTT is comparable. The next task was framed the same way as with the first one with PN but targeting a different system with the formal specification in GTT. Like the previous task, programme code is left out to remove the bias factor. The amount of writing is slashed to focus on the specific notation (for repetition description) as well as to harmonise with the other exam questions.

5.3 | Evaluating subjective user satisfaction (Experiment 3)

This experiment (Exp-2) was designed and planned to be conducted at the bachelor student level in perspective of behavioural complexity and the range of notation. The master students could be regarded as module developers who are able and willing to deal with this type of formal specification, while the bachelor students are comparable with the application engineers. Based on this reason, the subjects were decided to be master students in the previous experiments and then extended to the bachelor level in this experiment to conform to the appropriateness in an immature background level.

These are downsized and reduced to consider the level of the participants as well as to focus on the effectiveness of some specific notations of GTT. The ultimate goal of this experiment compared with the previous experiments is to achieve a qualitative assessment from the participants regarding their subjective perceptions. The experiment consisted of 40 min of tutorial and exercise tasks asked of participants to be answered for 35 min, including a qualitative questionnaire.

The exercises consisted of an understanding task and a creating task like the other experiments: answering the question regarding behaviour with respect to the brief natural language description and a specification in GTT and creating the specification based on the partially given specification in GTT. After that, the participants were asked to answer the evaluation questionnaire, which is a tailored version of the System Usability Scale (SUS) [54] regarding understanding, creating, and general impression. The questionnaire was intended to rate these aspects as perceptions in a Likert scale of one to five. Answer options consisted of the least to the most suitable of the questions including two negative, one neutral, and two positive answers, namely, *strongly disagree*, *disagree*, *uncertain*, *agree*, and *strongly agree*. To avoid unintentional factors, the statements were prepared in mixed-tone (positive and negative) by forcing the respondent to read each statement and make an effort to think about it (cf. [59, 60]). The overall experiment plan is summarised in Table 2.

6 | EXPERIMENT RESULTS

In the experiments, the participants participated in training and were asked to perform the exercise tasks and answer the subjective assessment questionnaire depending on the experiment. Additional questionnaires were given to the participants to collect their profiling information. Participants' profiles with regard to the understanding of their aspects are presented in Section 6.1. The results collected in the experiments are presented with descriptive and inference statistics in Section 6.2.

6.1 | Participants' profiles

One of the reasons for the conduct of these evaluations with university students is that they would be at a similar level to junior engineers [61] in the field as major future users of the language, that is, GTTs and that they are likely to use such types of languages in (future) practices. Therefore, bachelor students represent immature engineers with some basic theoretical knowledge about engineering and embedded system implementation, and master students are more mature in automation software engineering inclusive of specifications.

In Exp-1, the number of participants attending was 29, including 18 valid participants (9 for each group) regarding language training caused by the difference of timing. The overall profile is also effective for Exp-2 in which participants from the same group attended.

In Exp-2, there were 34 participants. Although it was not revealed clearly who had attended the tutorial due to the anonymity of the experiments, the participants could be regarded as fully trained since it is an exam that matters to their degree results. The participants were master course students with mostly a mechanical engineering major with an average grade of 2.26 (σ: 0.50) in the German grade mark scale with 1 as the highest and 5 as the lowest grade (Figure 2a). (Note: German grading system overview and comparable US/UK grading is shown in [62]) More than 80% of the participants had some experience in the industry in the form of internships and working. The work duration among the experienced ones

TABLE 2 Summary of the tasks performed in the experiments

Experiment		Task description	Time (min)	Target group[a]	Difficulty level	Remarks
Comparing effectiveness	**Exp-1 (N = 9)**[a]For Group1 and Group2 each	**1. Understanding specifications (1) in GTT/PN** To read the specification and find violation in the given shorter IEC 61131-3 code: checking branch condition and missing implementation	15	M	Base 1	▪ Given specification: Group1 in Generalise Test Table (GTT), Group2 in PN
		2. Understanding the specification (2) in GTT/PN–scaled To read the scaled specification and find violation in the given longer IEC 61131-3 code: checking branch condition	15		> base 1	▪ Given specification: Group1 in PN, Group2 in GTT (**switched** specification language) ▪ **Scaled** size specification and programme code targeting more difficult level
		3. Creating the specification To generate the specification regarding the natural language description and corresponding code in IEC 61131-3	30		Base 2	▪ Group1 in GTT → PN, Group2 in PN → GTT - **Conversed** creating order of the language ▪ More time assigned on creating
	Exp-2 (N = 34)	**1. Understanding the specification in PN** To read the specification with respect to the overall behaviour understanding and specific variable pair traces	6	M	< base 1 (base 3)	▪ Tasks are simplified by removing the code inspection part and balanced complexity-wise regarding the target system in GTT and PN questions
		2. Creating the specification in PN To implement partial repetition on the given PN (i) using a variable and (ii) adding more places	6		< base 2 (base 4)	▪ The order of solving tasks and time limit is not controlled
		3. Understanding the specification in GTTs To read the specification with respect to the overall behaviour understanding and specific variable pair traces	6		= base 3	- Marked on the left column is the expected time duration for each task
		4. Creating the specification in GTTs To implement partial repetition on the given GTT table	2		= base 4	- Participants would assign as they wish within the overall allowed time duration (90 min)

(Continues)

TABLE 2 (Continued)

Experiment		Task description	Time (min)	Target group[a]	Difficulty level	Remarks
Subjective satisfaction	**Exp-3 (*N* = 73)**	**1. Understanding the specification in GTTs**	15	B	= base 3	▪ Similar tasks to Exp-2 with more time considering the level of participants
		To read the specification with respect to the overall behaviour understanding and specific variable pair traces				
		2. Creating the specification in GTTs	10		< base 1 & > base 4	▪ Reduced scope in creating task compared with Exp-1; but most basic notations are included
		To implement partial repetition				
		3. Qualitative questionnaire	10		N/A	-

[a]M = Master, B = Bachelor.

spans from 2 to 8 months for an internship and from 6 to 48 months for part-time or full-time work (Figure 2b and 2c). The participants evaluated their programing skill as intermediate level (mean: 3.17 and σ: 0.17 in a Likert scale with 1 as the lowest and 5 as the highest) in a self-assessment.

In Exp-3, the participants were 73 bachelor students. Assuming that the students did not have influential industry experiences since the lecture was taken in the fifth semester of the bachelor course, only the grade information of the participants was gathered (Figure 2d).

6.2 | Experiment results and discussion

6.2.1 | The preliminary experiment (Exp-1)

First, in the understanding tasks (i.e., task1 and task2) of Exp-1, the participants were asked to find faults within the programme code with respect to the given specification in GTT or PN (Figure 3a, 3b). In the first task, GTT specification (Group1) showed better scores than PN one (Group2). Conversely, PN specification (Group1) showed a better result than GTT one (Group2). Although the overall point is higher in task2 for both languages, direct comparison between task1 and task2 would not be meaningful because of differences in the type of system behaviour, the size of specifications, and the length of the given code. Although the second task was targeted to evaluate the scaled case (i.e. more difficult), the PN specification scale was not managed well while GTT specifications and the given programme size were scaled. Therefore, it is hard to say which language is dominant over the other regarding understandability (H1.1).

For the creating task, the participants were asked to create specifications in both languages for one target system. The overall mean value of the results is very similar in GTT and PN (Figure 3c), and thus H1.2 is true. However, the results of each group are very interesting. For Group1, the specification creation task of GTT is given before PN, which is in reverse order in Group2. For each group, the preceding one (i.e. GTT for Group1 and PN for Group2) got a lower point. They probably learnt behaviour during the first specification creation task and could then represent the second specification creation better.

One observation of tasks is the sensitivity to the specification scale (cf. Table 3). Since the specification size grew around 2.3 times in GTT (from 3 to 7 rows), 25% degradation was observed. In PN, 37% degradation was observed while the specification size grew 2.3 times (from 8 to 18 edges). Although it is a simple comparison, not considering the difference of inputs/outputs that affect the GTT size, the sensitivity of the language to the growth of the specification seems less in GTT than in PN (H1.4).

The result was also analysed with respect to the background knowledge of the participants. The participant's profile, grade, working, and internship experience could be regarded as criteria for the background level. The trend line of GTT showed a more gentle slope than the one of PN (Figure 4a) even though it takes the position lower than that of PN. Here, the creation task result

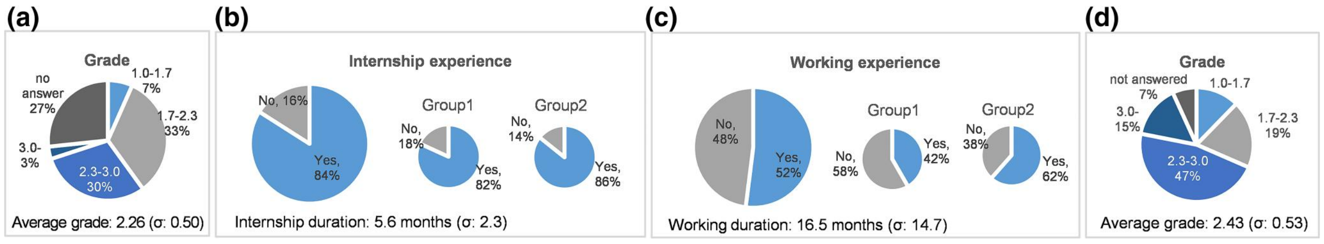


FIGURE 2 Profiles of the participants: (a)-(c) for Exp-1 and (d) for Exp-3

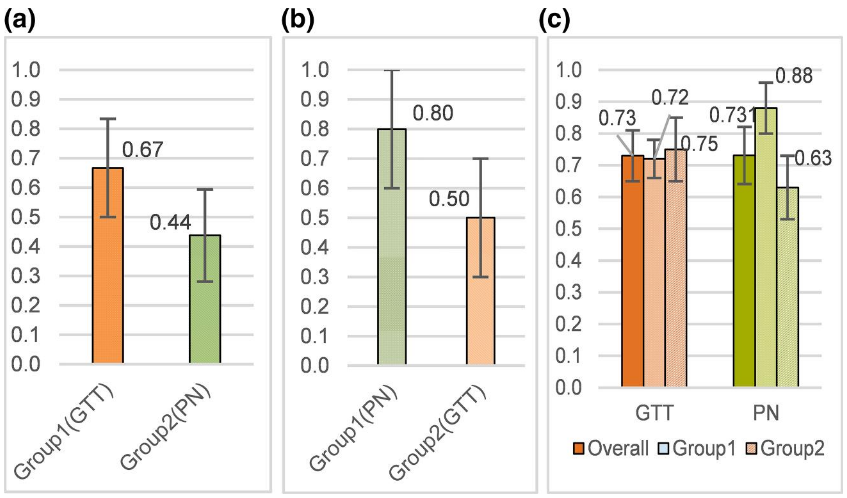


FIGURE 3 Exercise score comparison of Exp-1 (normalised mean value with error bars of the standard error): (a)–task1 (understanding), (b)–task2 (understanding, scaled), and (c)–task3 (creating)

was considered for comparison since the effective comparison is possible between GTT and PN with the valid number of samples who answered the task and shared the grade (6 for GTT and 8 for PN). The comparison of the mean value of creation results also showed that the internship experience is less influential in GTT than PN (Figure 4b). The comparison for working experiences showed no difference between GTT and PN (Figure 4c). Therefore, we conclude that H1.3 is true.

Since this was the first experiment, some bias points existed. We observed that some of the participants were not used to the IEC 61131-3 language. Although the implementation was comparably simple, this biased the result. Also, the delay between training and evaluation affected the controlling of attendance of the participants. Splitting the training session and evaluation, and non-compulsive evaluation resulted in a smaller number of valid participants. Besides, the fact that each group was given different tasks makes the comparison harder, especially due to the small number of subjects. Task1 and task2 were planned to see the effect of the scaled problem; however, the result could not be compared directly due to the low number of participants, which led to dependent results in the participants' profiles.

6.2.2 | Comparing effectiveness (Exp-2)

In this experiment, each participant had to solve two tasks (GTT and PN specification). The tasks are balanced and validated by transforming each specification from one language to the other. Scores are presented to be normalised to ease the comparison.

There were two points targeted in the understanding task: one was understanding the overall behaviour (allotted 8 points) and the other was tracking of variable value set changes (allotted 3 points). For both, GTT showed a statistically meaningful improvement (5.1% and 7.7%, respectively) and 5% for the overall score in comparison, supporting H1.1 to be true (Figure 5-a). In this task, it was required to describe the condition of the state change and the result of variables. Faults were mainly observed as the answer missed some necessary variables for some conditions and the results. It seemed that the variable names and the corresponding values are explicit in GTT for each state so that students missed less than in PN description where not all the variables are visible in all states and the condition descriptions.

Specification creating task was focussed on representing the repetition of certain behaviour. The block repetition notation of GTT was to be compared with that of PN. In the case of PN, two options were given: using a counter variable (easy) and adding places and corresponding edges (hard). It is hypothesised that participants would get higher scores by using a counting variable than by adding more PN notation elements (i.e. places and transitions) and thus was proved to be true as seen in the scores received by the students (Figure 5b). Comparing GTT with PN, students obviously got more points with the repetition block notation in GTT than by changing the net structure and slightly more than using the variable. Therefore, H1.2 is true. The

TABLE 3 Task and specification complexity comparison in Exp-1[a]

Group	Task	I/O	Specification size	Programme size	Score
(a) GTT					
Group1	Task1	2 IN/1 OUT	3 rows	18 lines	0.67
Group2	Task2	6 IN/4 OUT	7 rows	53 lines	0.50
(b) PN					
Group1	Task2	6 IN/4 OUT	5 places, 8 edges	53 lines	0.80
Group2	Task1	2 IN/1 OUT	5 places, 18 edges	18 lines	0.44

[a]The number of rows for Generalised Test Tables (GTT), and the number of places and edges for Petri Nets (PN) as specification size.

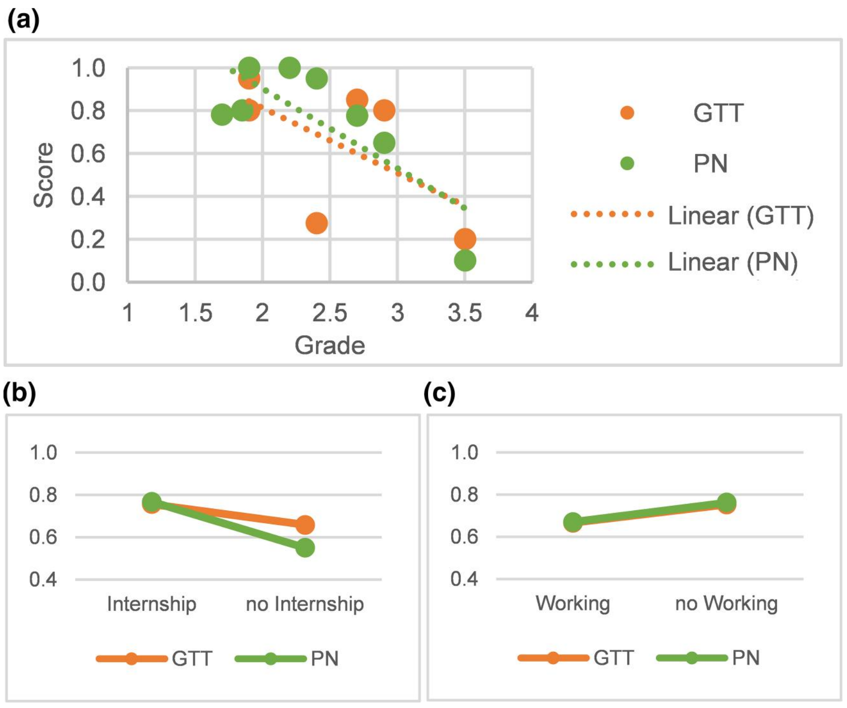


FIGURE 4 Correlation (based on Exp-1) of (a) score versus grade, (b) score versus internship experience, and (c) score versus working experience

simplicity of notation usages that are to be used to implement the required change might be affected. To indicate the repetition of the partial behaviours, GTT allows a specific block repetition notation with a repetition number indication simply in the duration column while the PN requires a proper notation usage of multiple elements, that is, places, transitions, and edges, to be a valid PN; and this difference could turn out to be the difference of the result.

The score of PN and GTT was related to the overall score, assuming that the total score indicates their background knowledge regarding software engineering. If one specification language required higher knowledge, the slope would be proportional to the overall score. As seen in Figure 5(c), GTT shows a more gradual slope than PN, which means that GTT has less correlation with the knowledge or background regarding software engineering. This supports H1.3 to be true.

6.2.3 | Subjective user satisfaction (Exp-3)

The focus of the third evaluation concerned the subjective perception of GTT. After the tutorial, participants were asked to solve exercise tasks and then to answer the questionnaire. The exercises were similar to the understanding task of Exp-2 but were simplified choice questions considering the different student levels. The score of the understanding task was 0.84 (normalised mean with standard deviation $\sigma = 1.53$ and skewness $\gamma_1 = -1.4$), which is slightly less than Exp-2 but could be an indication that the participants were following the concept successfully. The creating task was also adapted to the level by providing the table and filling the blanks. The mean score of the creating task was 0.78 ($\sigma = 3.57$, $\gamma_1 = -1.2$).

The scores were related to the grade of the participants assuming that the higher grade implies a higher knowledge level of software engineering. Consistent to the results of the

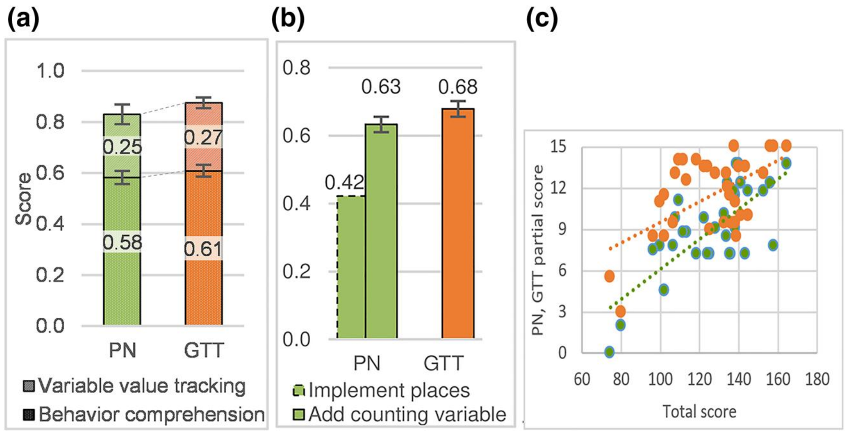


FIGURE 5 Score comparison of generalised Test Tables (GTT) and Petri Nets (PN) from Exp-2. (a) Understanding task, (b) Creating task, and (c) Correlation of score to the total score

previous experiments, the result of this experiment also supports the uncorrelated relationship between the score of the exercises and the overall grade (Figure 6), supporting H1.3.

For qualitative questions, each choice is rated as 1–most negative (i.e., strongly disagree), 2–negative (i.e., disagree), 3–neutral, 4–positive (agree), 5–most positive (i.e., strongly agree). The result of the negative tone question is presented as reversed in this paper for intuitive visualisation placing positives and negatives on each side. For example, for questions like 'Remembering the syntax was difficult,' disagreeing is actually positive. So the answer choices are reordered as five to one so that positive answers are always shown on the right side while negative answers are shown on the left.

The majority of the participants (81%) accept the language as an understandable concept for them (Figure 7a). Although the rate of comprehending the given task is lower than the understanding rate of the concept, more than half (58%) agreed that comprehending the specification in GTT is easy. In the case of remembering syntaxes, although the rating is spread somewhat more than with the others, the mean value is still above three, and the positive answer rate (44%) is higher than negative (25%). Therefore, the conclusion is still positive. Overall, as seen in the graph of the blue area and the skewness values, understandability can be deemed positive. Thus, H2.1 is true.

For creating (Figure 7b), more than half of the participants (56%) answered that they could apply GTT concepts to creating specifications. The result is lower than the understanding result, but it can still be considered as positive feedback. The participants answered that the given exercise tasks were not easy to create specifications (mean = 3.03), although they agreed that they could mostly understand the given task (39% positive, 31% neutral). The collected answers with difficulty points were mostly in writing Boolean expressions (15 out of 27 collected answers). Nevertheless, the participants agreed that they could create a specification in GTTs overall. Therefore, H2.2 is true.

Almost 65% of participants answered positively to usability of the GTT being easy (mean = 3.63, σ = 0.63, γ_1 = −0.73, Figure 7c). This is consistent with the result of the question regarding the perceived complexity in number 5, which was answered positively by 80% of participants. So, in general, the satisfaction rate seems to be high, which leads to H2.4 to be true. For learning, 77% of participants answered positive to the question that an application engineer can learn GTT in a reasonable time (a day or less)—strongest positive (mean = 3.89, σ = 0.67, γ_1 = −0.2), supporting H2.3 being true. However, the answer to the question regarding future usage is comparably lower than to the other questions, although the result is still positive with the mean of 3.33 (σ = 0.62) with γ_1 as 0.05. This supports H2.5 to be true.

7 | SUMMARY OF THE RESULT AND VALIDITY ANALYSIS

We have investigated the usability of the developed specification language, that is, GTT, through the execution of user evaluation and its analysis with regard to the RQs (summarised in Table 4).

For RQ1, we have hypothesised that users can understand the system behaviour with GTT similar to or better than with another conventional language (H1.1). In Exp-1 and Exp-2, the control language was chosen as PN. Although it was not revealed in Exp-1, GTT showed better results than PN regarding understanding the overall behaviour and variable value change tracking; and the explicit variable value representation in a structured format seemed to aid the understandability. Therefore, the result could be considered valid. For creating the specification, Exp-1 showed at least similar results in both languages, and Exp-2 showed the superiority of GTT with respect to representing the repetition of some specific block. This seemed to be supported by the simpler notations, especially for the repetition aspect. However, Exp-1 did not have a large enough number of participants that were not biased by differences of each and Exp-2 focussed on a specific notation only. Therefore, the validity level could be said to be weak, and the hypothesis should be investigated with a wider range of notation usages to support the hypothesis more strongly. Also, the correlation between usability and background of the user was also examined in the three experiments (H1.3). As seen consistently in the results, we

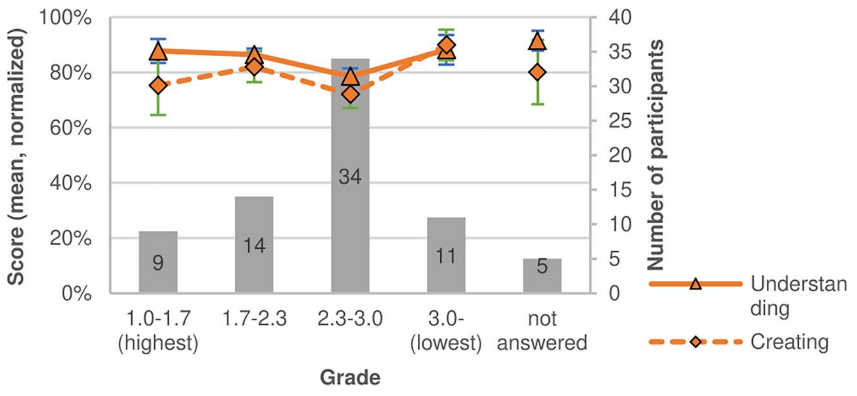
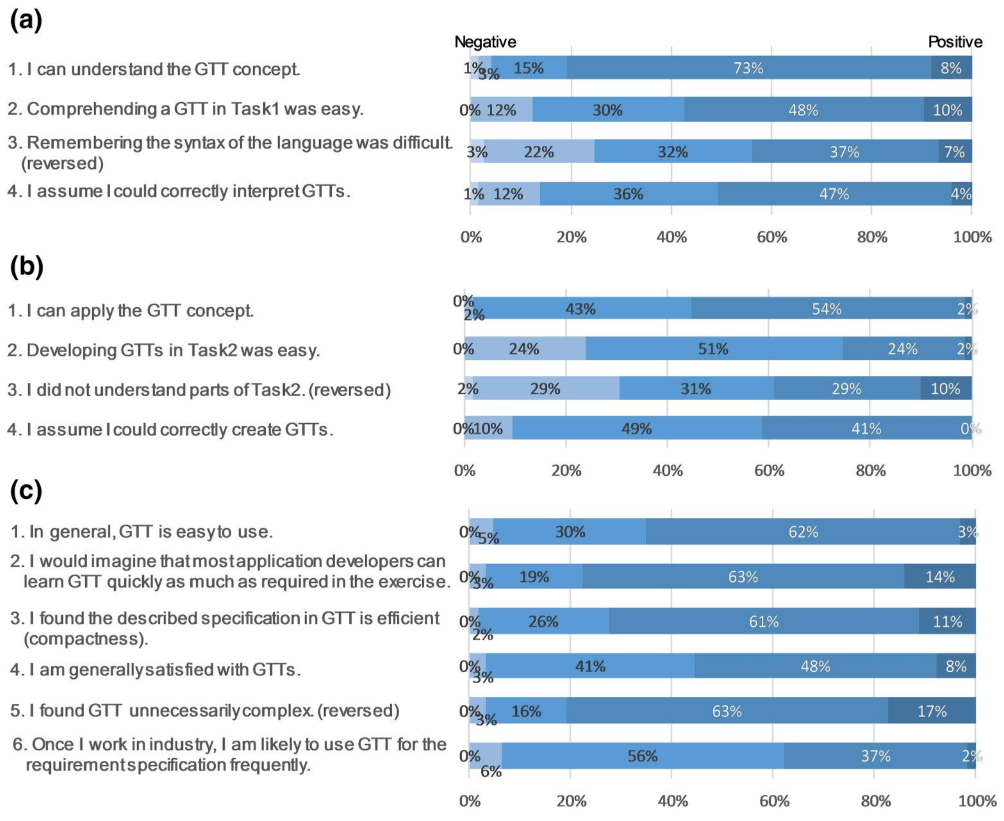


FIGURE 6 Correlation of the score and the grade from Exp-3



(a)					
1. I can understand the GTT concept.	1%	3%	15%	73%	8%
2. Comprehending a GTT in Task1 was easy.	0%	12%	30%	48%	10%
3. Remembering the syntax of the language was difficult. (reversed)	3%	22%	32%	37%	7%
4. I assume I could correctly interpret GTTs.	1%	12%	36%	47%	4%

(b)					
1. I can apply the GTT concept.	0%	2%	43%	54%	2%
2. Developing GTTs in Task2 was easy.	0%	24%	51%	24%	2%
3. I did not understand parts of Task2. (reversed)	2%	29%	31%	29%	10%
4. I assume I could correctly create GTTs.	0%	10%	49%	41%	0%

(c)					
1. In general, GTT is easy to use.	0%	5%	30%	62%	3%
2. I would imagine that most application developers can learn GTT quickly as much as required in the exercise.	0%	3%	19%	63%	14%
3. I found the described specification in GTT is efficient (compactness).	0%	2%	26%	61%	11%
4. I am generally satisfied with GTTs.	0%	3%	41%	48%	8%
5. I found GTT unnecessarily complex. (reversed)	0%	3%	16%	63%	17%
6. Once I work in industry, I am likely to use GTT for the requirement specification frequently.	0%	6%	56%	37%	2%

FIGURE 7 Qualitative evaluation result in Exp-3

found that the language is accepted comparably well with regard to the participants' grades and internship experiences. Although the examined result shows as hypothesised, it is under the assumption of the implying relationship between the background knowledge levels and the academic grades or internship experiences. Thus, the validity of this relationship should be clarified to validate the correlation analysis fully. The final hypothesis of RQ1 is the sensitivity of the specification scale (H1.4). We have concluded this to be true comparing the results from the tasks; however, the direct comparison between the groups in Exp-1 was not entirely sound. This is because on the one hand, the results are from different groups, and on the other hand, the groups lack representativeness due to the limited number of participants. Also, the scale of the specifications and of the established problem, including the described system, was not well enough managed to be able to verify the hypothesis. Thus, the result cannot be regarded as valid.

Speculating about the subjective perception levels of the language users for RQ2, we have hypothesised a positive user perception level for understanding and creating activities (H2.1, H2.2). As the result shows, we got a high rate of positive feedback on both (higher in understanding). Users also regarded the language to be reasonably learnable (H2.3). However, for this point, additional evaluation regarding learning would be necessary to measure the precise level of

TABLE 4 Evaluation of the research question (RQ) and hypotheses—summary

RQ	Related hypothesis	Result (validity)	Related experiment
RQ1: Is the target language effective for the purpose?	H1.1: A user can **understand** a system behaviour through the specification in GTTs similar to or better than through the conventional language (Petri Nets [PN]).	True (valid)	Exp-1 (Figure 3a, 3b), Exp-2 (Figure 5a)
	H1.2: A user can **create** a specification regarding a system behaviour in Generalised Test Tables (GTTs) similar to or better than in the conventional language (PN).	True (weakly valid)	Exp-1 (Figure 3c), Exp-2 (Figure 5b)
	H1.3: **Understanding** and **creating** specifications in GTTs is less related to the **background knowledge** of the user regarding software engineering than conventional language (PN).	True (weak valid)	Exp-1 (Figure 4), Exp-2 (Figure 5c), Exp-3 (Figure 6)
	H1.4: The score of **understanding** task is less sensitive to the **scale** of the specification in the target language than the conventional language (PN).	True (not valid)	Exp-1 (Table 3)
RQ2: Is the target language satisfactory to automation engineers?	H2.1: Users have confidence in understanding a specification in GTTs.	True (valid)	Exp-3 (Figure 7a)
	H2.2: Users have confidence in creating a specification in GTTs.	True (valid)	Exp-3 (Figure 7b)
	H2.3: Users think the language, that is GTTs, is reasonably learnable.	True (valid)	Exp-3 (Figure 7c #2)
	H2.4: Users are satisfied with the language, that is GTTs, as a behaviour specification representation method in general.	True (valid)	Exp-3 (Figure 7c)
	H2.5: Users would like to use the language, that is GTTs, as a behaviour specification representation method in the future.	True (weakly valid)	Exp-3 (Figure 7c #6)

learnability. While the overall user perception regarding its complexity and satisfaction was also positive (H2.4), we also found that almost half of the participants were still wavering over whether to use it in practice in spite of positive feedback on the average (also observable as $\gamma_1 = -0.20$–the lowest among all the questions). Therefore, performing successive work to figure out the main reason for this phenomenon and the corresponding adaptation of the language would be necessary. In addition, to make the results of the qualitative assessment stronger, a further survey comparing other formal specification languages with a promising number of participants should be conducted.

Through the evaluation, some potential improvement points were also revealed. Although it is easy to follow the tabular structure, some value changes are misread or overlooked when cells are filled with text values. This could be improved by putting a change indication notation (e.g. bold font or colouring in the text or cell). Also, as seen in Exp-1, and between Exp-1 and Exp-2, scaling affected the degradation of the score. Scaling should be supported further by, for example, additional notation or restructuring so that the more variable or longer behaviour could be comprehended easier; and this might allow representing some conditional branching which is not possible through the current GTT notations by nature of the tabular structure.

8 | CONCLUSION AND OUTLOOK

Domain-specific languages are being developed to solve or ease the problems in system engineering activities and are definitely helpful for their target problem-solving. However, not all of the developed languages are widely used, and one of the main reasons is the absence of proof of their functionality and usability, which can be obtained through appropriate studies and evaluations. Targeting the prevalence of formal verification in aPS engineering, a formal specification language based on industry practices has been developed, namely generalised test tables. We studied its functional applicability previously in parallel with developing its applications in several methods, and this is followed by the usability study as presented in this paper.

By answering the RQs with hypotheses and verifying them through experiment evaluations, various usability aspects of the developed language have been declared. However, since one of the major motivations of the language was the ease of the user who would be the main control software developer and tester, a more precise usability study and the corresponding evolution of the language would be mandatory for it to be more powerful. Based on the result of the experiments, a comparative subjective assessment would be beneficial to position the relative perception level. Also, usability studies with industry personnel, who would need and use the language for their daily activities, are compulsory to accelerate usage of the language aimed at the ultimate prevalence of formal verification.

To obtain additional eligibility, a more specific evaluation with regard to the standard such as [63] would be necessary to discover the points to be developed and elaborated to a sound specification language. Since everyday usage of the language would be with a corresponding tool, such as an integrated development environment or a testing tool, a tool-based usability study should also be conducted to assess usability of the language with the tool as well as the tool usability itself.

ACKNOWLEDGEMENT

This work was supported by the German Research Foundation (DFG) under the Priority Programme 1593: Design For Future —Managed Software Evolution (grant number: VO 937/28-2, BE2334/seven to two and UL 433/1-2).

ORCID

Suhyun Cha https://orcid.org/0000-0001-7477-8008
Birgit Vogel-Heuser https://orcid.org/0000-0003-2785-8819

REFERENCES

1. Vogel-Heuser, B., et al.: Evolution of software in automated production systems: challenges and research directions. J. Syst. Software. 110, 54–84 (2015)
2. Vogel-Heuser, B., Hess, D.: Guest editorial industry 4.0-prerequisites and visions. IEEE Trans. Automat. Sci. Eng. 13(2), 411–413 (2016)
3. Thramboulidis, K.: The 3+1 SysML view-model in model integrated mechatronics. JSEA. 03(2), 109–118 (2010)
4. Vogel-Heuser, B., Ocker, F.: Maintainability and evolvability of control software in machine and plant manufacturing–an industrial survey. Contr. Eng. Pract. 80(September), 157–173 (2018)
5. Zonnenshain, A., Kenett, R.S.: Quality 4.0-the challenging future of quality engineering. Qual. Eng. 32(4), 614–626 (2020)
6. Ulewicz, S., et al.: Proving equivalence between control software variants for programmable logic controllers. In: Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation, pp. 1–5 (2015)
7. Rösch, S., Vogel-Heuser, B.: A light-weight fault Injection approach to test automated production system PLC software in industrial practice. Contr. Eng. Pract. 58(Jan.), 12–23 (2017)
8. Cha, S., et al.: Applicability of generalized test tables: a case study using the manufacturing system demonstrator xPPU. Automatisierungstechnik. 66(10), 834–848 (2018)
9. Frey, G., Litz, L.: Formal methods in PLC programing. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 2431–2436 (2000)
10. Sinha, R., et al.: A survey of static formal methods for building dependable industrial automation systems. IEEE Trans. Ind. Inf. 15(7), 3772–3783 (2019)
11. Ljungkrantz, O., et al.: An empirical study of control logic specifications for programmable logic controllers. Empir. Software Eng. 19(3), 655–677 (2014)
12. Pang, C., et al.: A study on user-friendly formal specification languages for requirements formalisation. In: Proceedings of IEEE International Conference on Industrial Informatics, pp. 676–682 (2016)
13. Adiego, B.F., et al.: Applying model checking to industrial-sized PLC programs. IEEE Trans. Ind. Informatics. 11(6), 1400–1410 (2015)
14. Holzmann, G.J.: The logic of bugs. SIGSOFT Softw. Eng. Notes. 27, 81–87 (2002)
15. Pakonen, A., et al.: User-friendly formal specification languages–conclusions drawn from industrial experience on model checking. In: Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation, pp. 1–8 (2016)
16. Weigl, A., et al.: Generalized test tables: a powerful and intuitive specification language for reactive systems. In: Proceedings of IEEE International Conference on Industrial Informatics, pp. 875–882 (2017)
17. Beckert, B., et al.: Generalised test tables: a practical specification language for reactive systems. In: Proceedings of International Conference on integrated Formal Methods, pp. 129–144 (2017)
18. Rösch, S.: Model-based testing of fault scenarios in production automation. PhD thesis. Technische Universität München, Munich, Germany (2016)
19. Ulewicz, S., Vogel-Heuser, B.: Automatisiertes Testen von Sondermaschinen – von der Modulbibliothek bis zur Anlage (en: automated testing of special purpose machines–from the module library to the plant). pp. 53–65.Tagungsband Automation Symposium, Dusseldorf, Germany (2015)
20. Barišic, A., et al.: Evaluating the usability of domain-specific languages formal and practical aspects of domain-specific languages, pp. 386–407. IGI Global, Hershey, USA (2014). https://doi.org/10.4018/978-1-4666-2092-6
21. van Deursen, A., Klint, P.: Domain-specific language design requires feature descriptions. J. Comput. Inform. Tech. 10(1), 1–17 (2002)
22. Rodrigues, I.P., Campos, D.B.: Usability evaluation of domain-specific languages: a systematic literature review, Human-Computer Interaction. User Interface Design, Development and Multimodality, pp. 522–534. Springer, Cham (2017)
23. Challenger, M., Kardas, G., Tekinerdogan, B.: A systematic approach to evaluating domain-specific modelling language environments for multi-agent systems. Software Qual. J. 24(3), 755–795 (2016)
24. Coursaris, C.K., Kim, D.J.: A qualitative review of empirical mobile usability studies. In: Association for Information Systems 12th Americas Conference on Information System (AMCIS), January 2006, vol. 2006(5), pp. 2873–2879 (2006)
25. International Organization for Standardization (ISO): ISO 9241-210:2010 Ergonomics of human-system interaction—Part 210: human-centred design for interactive systems (2010)
26. Schalles, C.: A framework for usability evaluation of modelling languages. Usability evaluation of modelling languages: an empirical research study, pp. 43–68, Springer Gabler, Wiesbaden (2013)
27. International Organization for Standardization (ISO): ISO/IEC 25010 Systems and software engineering—systems and software Quality Requirements and Evaluation (SQuaRE) — system and software quality models (2011)
28. Bevan, N.: Measuring usability as quality of use. Software Qual. J. 4(2), 115–130 (1995)
29. Moon, I., et al.: Automatic verification of sequential control systems using temporal logic. AIChE J (1992)
30. Shatrov, V., Vyatkin, V.: Formal verification of IEC 61499 enhanced with timed events. In: Proceedings of Advanced Doctoral Conference on Computing, Electrical and Industrial Systems, pp. 168–178 (2020)
31. Bauer, N., et al.: Verification of PLC programs given as sequential function charts. In: Integration of Software Specification Techniques for Applications in Engineering: Priority Program SoftSpez of the German Research Foundation (DFG), Final Report, pp. 517–540. Springer Berlin Heidelberg (2004)

32. Biallas, S., Brauer, J., Kowalewski, S.: Arcade.PLC: a verification platform for programmable logic controllers. In: Proceedings of IEEE/ACM International Conference on Automated Software Engineering, pp. 338–341 (2012)
33. Wardana, A.N.I., Folmer, J., Vogel-Heuser, B.: Automatic program verification of continuous function chart based on model checking. In: Proceedings of 35th Annual Conference on Industrial Electronics, pp. 2422–2427 (2009)
34. Ren, H., et al.: Verification using counterexample fragment based specification relaxation: case of modular/concurrent linear hybrid automata. IET Cyber-phys. Syst. 2(2), 65–74 (2017)
35. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. Proc. Work. Form. Methods Softw. Pract. 411–420 (1999)
36. Campos, J.C., Machado, J.: Pattern-based analysis of automated production systems. IFAC Proc. 13(1), 972–977 (2009)
37. Fockel, M., et al.: Formal, model- and scenario-based requirement patterns. In: Proceedings of International Conference on Model-Driven Engineering and Software Developement, pp. 311–318 (2018)
38. DIN:DIN EN 60848: GRAFCET specification language for sequential function charts (2014)
39. Julius, R., et al.: A meta-model based environment for GRAFCET specifications. In: Proceedings of International Systems Conference, pp. 1–7 (2019)
40. Vyatkin, V., Bouzon, G.: Using visual specifications in verification of industrial automation controllers. EURASIP J. Embed. Syst. 2008(1), 1–9 (2008)
41. Darvas, D., MajzikIstván, I., Viñuela, E.B.: Well-formedness and invariant checking of PLCspecif specifications. In: Proceeding of the 23rd PhD Mini-Symposium of the Budapest University of Technology and Economics, Department of Measurement and Information Systems, pp. 10–13 (2017)
42. Przigoda, N., et al.: Verifying the structure and behavior in UML/OCL models using satisfiability solvers. IET Cyber-phys. Syst. 1(1), 49–59 (2016)
43. Ruiz, J., Serral, E., Snoeck, M.: Evaluating user interface generation approaches: model-based versus model-driven development. Softw. Syst. Model. 18(4), 2753–2776 (2019)
44. Teruel, M.A., et al.: A CSCW requirements engineering CASE tool: development and usability evaluation. Inf. Software Technol. 56(8), 922–949 (2014)
45. Snook, C.F., Harrison, R.: Experimental comparison of the comprehensibility of a Z specification and its implementation in Java. Inf. Software Technol. 46(14), 955–971 (2004)
46. Carew, D., Exton, C., Buckley, J.: An empirical investigation of the comprehensibility of requirements specifications. In: Proceedings of International Symposium on Empirical Software Engineering, pp. 256–265 (2005)
47. Razali, R., et al.: Experimental comparison of the comprehensibility of a UML-based formal specification versus a textual one. In: Proceedings of International Conference on Evaluation and Assessment in Software Engineering, pp. 1–11 (2007)
48. Bitsch, F.: Safety patterns - the key to formal specification of safety requirements. Comput. Safety, Reliab. Secur. (1064), 176–189 (2001)
49. Tenbergen, B., Weyer, T., Pohl, K.: Hazard Relation Diagrams: a diagrammatic representation to increase validation objectivity of requirements-based hazard mitigations. Requirements Eng. 23(2), 291–329 (2018)
50. Khalajzadeh, H., et al.: BiDaML: a suite of visual languages for supporting end-user data analytics. In: Proceedings of the 2019 IEEE International Congress on Big Data, pp. 93–97 (2019)
51. CODESYS GmbH: CODESYS store international - CODESYS test manager. https://store.codesys.com/codesys-test-manager.html?___store=en. Accessed February 2021
52. Zeiss, B., Vega, D.: Applying the ISO 9126 quality model to test specifications. Softw. Eng. 2007(105), 231–244 (2007)
53. Ludewig, J.: Languages, methods, and tools for software specification'. Hardware and software for real time process control, pp. 225–256 (1989)
54. Brooke, J.: SUS-A quick and dirty usability scale. Usability Eval. Ind. 189(194), 4–7 (1996)
55. David, R., Alla, H.: Petri nets and grafcet: tools for modelling discrete event systems. Prentice Hall, Upper Saddle River, USA (1992)
56. Vogel-Heuser, B., et al.: Fault handling in PLC-based industry 4.0 automated production systems as a basis for restart and self-configuration and its evaluation. J. Softw. Eng. Appl. 09(1), 1–43 (2016)
57. Spindler, M., et al.: Erstellung von steuerungssoftware für automatisierte materialflusssysteme per drag & drop (en: engineering the control software of automated material handling systems via drag & drop). Logist J 2017, 9–14 (2017)
58. Shanahan, T., Lomax, R.: A developmental comparison of three theoretical models of the reading-writing relationship. Res. Teach. English. 22(2), 196–212 (1988)
59. Davis, F., D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. 13(3), 319 (1989)
60. Wong, N., Rindfleisch, A., Burroughs, J.E.: Do reverse-worded items confound measures in cross-cultural consumer research? The case of the material values scale. J. Consum. Res. 30(1), 72–91 (2003)
61. U.S. Bureau of Labor Statistics: Mechanical engineers–occupational outlook handbook. https://www.bls.gov/OOH/architecture-and-engineering/mechanical-engineers.htm. Accessed July 2020
62. Studying in Germany: 'German grading system', https://www.studying-in-germany.org/german-grading-system/. Accessed July 2020
63. IEEE: IEEE 1850-2010 - standard for property specification language (PSL) (2010)

How to cite this article: Cha, S., et al.: Table-based formal specification approaches for control engineers—empirical studies of usability. *IET Cyber-phys. Syst.*, Theory Appl. 1–15(2021). https://doi.org/10.1049/cps2.12017