Assessing the impact of hints in learning formal specification

Anonymous Author(s)

ABSTRACT

Background: Many programming environments include automated feedback in the form of hints to help novices learn programming autonomously. Some experimental studies investigated the impact of automated hints in the immediate performance and learning retention in that context. Automated feedback is also becoming a popular research topic in the context of formal specification languages, but so far no experimental studies have been conducted to assess the impact of hints while learning such languages. *Objective*: We aim to investigate the impact of different types of automated hints while learning a formal specification language, not only in terms of immediate performance and learning retention, but also in the emotional response of the students. *Method*: We conducted a simple one-factor randomised experiment in 2 sessions involving 85 undergraduate students majoring in computer science and engineering. In the 1st session students were divided in one control group and three experimental groups, each receiving a different type of hint while learning to specify simple requirements with the Alloy formal specification language. To assess the impact of hints on learning retention, in the 2nd session, 1 week later, all students had no hints while formalising requirements. Before and after each session the students answered a standard self-reporting emotional survey to assess their emotional response to the experiment. *Results*: Of the three types of hints we evaluated, only the ones that point to the precise location of an error had a positive impact on the immediate performance and none of them had significant impact in the learning retention. Hint availability also causes a significant impact on the emotional response, but no significant emotional impact exists once hints are no longer available (e.g. no significant deprivation effects were detected). *Conclusion*: Although none of the evaluated hints had an impact on learning retention, learning a formal specification language with an environment that provides hints with precise error locations seems to contribute to a better overall experience without apparent drawbacks. Further studies are needed to investigate if other kinds of hints, namely hints combined with some sort of self-explanation prompts, can have a positive impact in learning retention.

CCS CONCEPTS

• **Software and its engineering → Specification languages**; • **Social and professional topics → Software engineering education**; • **Applied computing → Computer-assisted instruction**.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Learning formal specification, automated hints, user study, Alloy

ACM Reference Format:
Anonymous Author(s). 2018. Assessing the impact of hints in learning formal specification. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Tutoring systems have long employed automated feedback to support students learning to program, namely help them assess the correctness of their programs and suggest hints to fix incorrect ones. Naturally, several studies have been performed to evaluate the impact of such hints on learning (sometimes with contradictory conclusions). Formal specifications are starting to permeate various software engineering techniques (such as automated testing, code synthesis, or contract programming) but tutoring systems and automated feedback for learning formal specification are still rare.

Alloy [17, 18] is a formal specification language popular in introductory courses to formal methods in software engineering[1]. It combines relational and temporal logic, and is supported by an Analyzer[2] that automates the generation of scenarios and the verification of properties. Relational logic extends first-order logic with operators that simplify the specification of structural properties using a navigational style resembling object-oriented programming. Temporal logic was a recent addition to this language [24], allowing the specification of the behaviour of systems and their expected temporal requirements using a linear time logic akin to TLA$^+$ [20].

Recently, the Alloy4Fun online platform was developed to serve as an auto-grader for teaching Alloy [26]. Recent advances on topics such as specification repair or principled scenario generation will soon enable the deployment of automatic hints in such a platform. However, there is some evidence that both students and experienced practitioners struggle to write correct formal specifications with Alloy [28], and that adding automatic feedback to the process might not be effective [11, 12]. In fact, it is known that the human aspects of formal methods are mostly unexplored [19].

It is unclear if the results from the studies assessing the impact of hints in learning programming also apply when learning to write formal specifications. Moreover, such studies mostly focus on test performance and not on the role of emotions in the educational context. Emotions can have an impact on memory [10, 23, 35], attention and motivation [23, 35], academic performance/learning process [3, 14, 35, 40], or cognitive task performance [14, 35], which is further underlined by recent neuroscience research addressing the neurological processes that formalise the link between emotion and cognition [23]. Given the importance of this topic, standard (and well-validated) emotional response assessment tools and questionnaires should be employed in education related studies .

[1] http://alloytools.org/citations/courses.html
[2] http://alloytools.org/

This work is a first step towards investigating the impact of automated feedback in learning formal specification. More specifically, we present the first user study assessing the impact of automated hints in learning the Alloy formal specification language, and the first to include an analysis of the emotional response to hints using standard questionnaires. Our experiment tested whether different types of hints have an impact on the immediate performance while learning, on the future performance of students while specifying requirements without hints (learning retention), and on their emotional response during the whole process. We considered 3 types of hints that could be automated in a platform like Alloy4Fun: (1) highlighting error locations, (2) providing visual counter-examples, and (3) providing natural language descriptions of incorrectly specified requirements. We conducted a simple one-factor randomised experiment in 2 sessions involving 85 undergraduate students majoring in computer science and engineering. In the 1st session participants with no Alloy background were asked to fix incorrect specifications while being supported by different types of hints (with a control group without hints); a week later, the participants were asked to specify requirements from scratch without any help of hints. We provide a replication package so that the experiments and the analysis can be replicated.

The remainder of this paper is structured as follows. Section 2 presents related studies assessing the impact of automated hints in learning. Section 3 presents the design of our experience. The results are presented in Section 4. Section 5 presents threats to the validity of our experiment, followed by a discussion of the results in Section 6. Section 7 wraps up the paper with a brief conclusion.

2 RELATED WORK

This section explores existing work on the impact of automated feedback in CS learning.

Next-step hints. A popular type of feedback provided by automated tutoring systems are next-step hints [32], that given an incorrect program try to guide the student towards a solution. Strategies to obtain these hints include data-driven approaches based on previous student attempts, search-based approaches that produce edit sequences to reach a solution, or relying on experts' annotations of known problematic patterns. We are not aware of any next-state hint technique for formal specifications, but they could be implemented using existing automatic repair techniques given an oracle written by the lecturers. For instance, there are techniques to fix Object Constraint Language (OCL) constraints given valid model instances [8], and Alloy repair techniques have been proposed that use as oracles test cases [41], assertions [4, 5, 45] or predicates [6].

Despite their popularity, research on the impact of next-step code hints in learning has been contradictory (although studies often use different learning environments and different types of hints). In the context of learning Lisp, a variety of feedback types and feedback delivery timings improved immediate performance and future performance in a post-test [9]. Data-driven hints for Python have shown no impact on learning (correct attempts at first try in post-tests) and only marginal positive impact in immediate performance (solving speed) [38], but also no significant impact in immediate performance (attempts until a correct one) nor in learning (performance in post-tests) [36]. Expert-authored hints for common mistake patterns in BlockPy resulted in better performance (test score) in the beginning of the semester, but the benefits faded out by the end of the semester [15]. When learning Prolog, both automatically highlighting problematic patterns and additionally providing expert-curated feedback improved immediate performance, and the latter also reduced the number of incorrect attempts (but not solving time) [22]. Known causes for limited learning effects when using this kind of feedback include hint abuse and avoidance by the students [1] and the fact that they drive the students to focus more on 'how' to fix errors than 'why' [29]. Although some of these studies were complemented with questionnaires, none used standard tools to measure emotional response.

Counter-examples as hints. As far as we are aware, Alloy4Fun is the only platform that has been designed to support the learning of formal specifications. The Alloy Analyzer – and Alloy4Fun – can generate instance scenarios for validation and counter-example for broken expected properties. These are presented graphically, and the visualisation can be customised by users. Alloy exercises in Alloy4Fun are usually checked by semantic equivalence against an oracle specification [26], so students naturally get a counter-example for incorrect attempts. Despite being one of the most popular features of Alloy, and a preliminary study showing that scenarios may improve the understanding of formal specifications [12], there is also evidence suggesting novices struggle to interpret instances and counter-examples [28]. In [26] the authors have introduced additional information in the counter-examples to help students interpret them. Techniques for guided instance generation [25, 34, 44] could improve the quality of this kind of feedback, but a preliminary study with students has shown mixed results [11].

In learning programming, this kind of feedback has similarities with auto-grading platforms that automatically generate test cases as evidence that attempts are not yet correct. Despite the concern for over-reliance on this kind of feedback [2], a recent study has shown that it improved immediate performance of students learning Java, and that their future performance was not affected negatively [33].

Enhancing hints with explanations. Other approaches have focused on promoting code comprehension by the students. Approaches include providing (and eliciting) explanations, and visualisations of executions, with varied results [16].

In the block-based programming environment Snap!, data-driven next-step hints enhanced with expert-authored explanations showed no significant impact on immediate or future performance (completed tasks) [29]. However, a different study showed that next-step hints with and without prompts for self-explanation improved immediate performance, but that only hints with self-explanation improved the performance in future tasks [31]. A conceptual replication study concluded that hints supported by explanations and self-explanation prompts improved efficiency (completed tasks and time spent), but the impact on future tasks was inconclusive [30]. When learning Python, a study has compared the performance of students when provided with FAQ-like hints with and without reflection prompts, concluding that the combination with reflection prompts had an immediate and long term impact on performance (assignment score), without decreasing efficiency (number of attempts) [7]. A different approach, also for Snap!, relied instead on worked examples, a demonstration of how other, similar exercises

are solved. A qualitative study showed students found it more helpful than next-step hints [42]. A study comparing the performance of students learning Python supported only by test cases, test cases accompanied by next-step hints, and test cases supported by trace visualisations showed that only the next-step hints improved immediate performance, and that problem comprehension was actually lower when supported by the trace visualisations [37].

Despite the possible positive impact on learning, many of these techniques require experts to write explanations. Recent advances in LLMs have already shown to be partially successful in explaining code snippets [27], which could potentially automate this process in the educational context. As far as we know, no work has been published on using explanations to help fixing formal specifications.

3 EXPERIMENTAL SETTING

This section presents our research goal and describes in detail the experimental setting.

3.1 Goal

Analyse different types of hints for fixing formal specifications
For the purpose of comparing their effect
With respect to immediate performance, learning retention, and emotional response
From the point of view of researchers
In the context of graduate level students majoring in computer science.

3.2 Procedure

The factor under investigation is the type of hint provided to students for fixing an incorrect specification, and we examined it with a control level and 3 experimental levels, corresponding to 3 different kinds of automatable hints. More detail about the tasks, the different types of hints, and their instrumentation in Alloy4Fun is given in Sections 3.6 and 3.7. To analyse the impact of these different hints we conducted a simple one-factor randomised experiment in the last two weeks of March 2023, in which the treatment administered to each participant corresponds to one of the 4 levels.

Since our goal was to assess the impact not only on the immediate performance, but also on learning retention, we divided the experiment in 2 sessions: in the 1st session, where participants first learned about Alloy, the different formal specification tasks were aided by hints (except for the control group); in the 2nd session, 1 week later, all participants attempted to solve formal specification tasks without hints. In both sessions the participants answered a standard emotional questionnaire (more details later) before and after the experiment to measure their emotional response. An overview of the research procedure is presented in Figure 1.

The 1st session proceeded with the following steps:

(1) Researchers greeted the participants and explained the overall research objective without disclosing that each participant would receive a different treatment. Each participant was randomly assigned to a group by drawing from a pool an unique identification code (ID) whose last character identified the treatment level. Subsequently, the participants were directed to 6 different physical rooms, each mixing participants with different treatments.

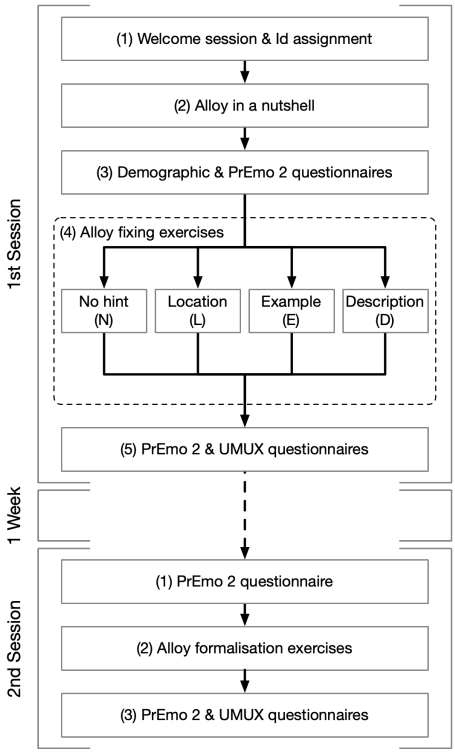


Figure 1: Overview of the experimental procedure

(2) A 1-page "Alloy in a Nutshell" guide to the language was provided to each participant, which they had around 10min to study and that they could consult during the tasks.
(3) A link to the demographic questionnaire was provided, which included questions such as the given ID, age, gender, familiarity with the Alloy language, and average academic grade. At the end of the demographic questionnaire participants were redirected to the emotional questionnaire.
(4) After the questionnaires, the participants were redirected to an Alloy4Fun page with the 1st of 12 tasks (described in Section 3.6). Each task had a maximum timeout of 5min to complete (the maximum duration of the experiment was thus 60min). After successfully finishing each task, or after the timeout, participants had to proceed to the next task. The Alloy4Fun page for each task had different hints according to the assigned treatment (detailed in Section 3.7).
(5) After the last task the students were redirected again to the emotional questionnaire, and, after completing that one, to a standard User Experience (UX) questionnaire.

The 2nd session proceeded 1 week later with the following steps:

(1) Participants answered the emotional questionnaire.
(2) At the end of the questionnaire the participants were redirected to an Alloy4Fun page with the 1st of 12 tasks, with no hints provided to any of the groups. The tasks had the same 5min timeout (maximum running time was again 60min), and this time no hints where provided to any of the groups. Participants could still consult the 1-page Alloy sheet.

(3) After the last task the students were again redirected to the emotional and UX questionnaires.

The provided UX questionnaire was **UMUX** [13], a standard questionnaire that measures system usability through 4 questions on a 7-point Likert scale. For the emotional questionnaire we resorted to **PrEmo 2** [21], a pictographic tool composed of 14 images depicting different categorical emotions (7 positive and 7 negative), that can be used to evaluate participants' self-reported emotional feelings towards a task and an experiment as a whole. A score for each emotion was reported on a 5-point Likert scale. An open-ended question box was also provided, allowing participants to provide justifications or additional comments.

3.3 Variables

The only independent variable of our experiment is the *Type of Hint* (HINT), operationalised by giving students different types of hints while learning formal specification in the 1st session. We have a control level with no-hints (N) and 3 experimental levels, corresponding to 3 different kinds of automatable hints: (L) the precise location of the error in a wrong specification, (E) a counter-example that illustrates why a specification is wrong, and (D) a natural-language description of a wrong specification.

We have three dependent variables:

- *Immediate performance* while learning with different kinds of hints in the 1st session, operationalised with two performance metrics: PROD_1, the immediate productivity, measured as the number of tasks successfully completed in the 1st session (ranging from 0 to 12), and EFF_1, the immediate efficiency measured as the ratio between the number of tasks successfully completed and the total number of attempts to solve all the 12 tasks in the 1st session.
- *Learning retention* while performing similar tasks without hints in the 2nd session, operationalised with similar performance metrics: PROD_2, the future productivity, and EFF_2, for future efficiency.
- *Emotional response* during the learning process, operationalised by two metrics aggregating the differential in the self-reported emotional feelings in the PrEmo 2 surveys before and after both sessions: EMO_1, the immediate emotional response, measured as the sum of differential in positive emotions minus the sum of differential in negative emotions during the 1st session, and EMO_2, the future emotional response, measured similarly. For each emotion we only measured if there was a variation in any direction (-1 to +1), so these aggregates vary in a scale from -14 to +14.

3.4 Hypotheses

We formulated 6 hypotheses, one for each of the above operational variables. The null hypotheses are formulated as follows:

- $H_0^{\mathsf{PROD}_1}$ – The type of hint does not have an effect on immediate productivity.
- $H_0^{\mathsf{EFF}_1}$ – The type of hint does not have an effect on immediate efficiency.
- $H_0^{\mathsf{PROD}_2}$ – The type of hint does not have an effect on future productivity.
- $H_0^{\mathsf{EFF}_2}$ – The type of hint does not have an effect on future efficiency.
- $H_0^{\mathsf{EMO}_1}$ – The type of hint does not have an effect on immediate emotional response.
- $H_0^{\mathsf{EMO}_2}$ – The type of hint does not have an effect on future emotional response.

The formulation for the alternative hypotheses is the following:

- $H_1^{\mathsf{PROD}_1}$ – The type of hint has an effect on immediate productivity.
- $H_1^{\mathsf{EFF}_1}$ – The type of hint has an effect on immediate efficiency.
- $H_1^{\mathsf{PROD}_2}$ – The type of hint has an effect on future productivity.
- $H_1^{\mathsf{EFF}_2}$ – The type of hint has an effect on future efficiency.
- $H_1^{\mathsf{EMO}_1}$ – The type of hint has an effect on immediate emotional response.
- $H_1^{\mathsf{EMO}_2}$ – The type of hint has an effect on future emotional response.

All the hypotheses are formulated as two-tailed since no theoretical or empirical support exists to presume a direction for the effect. As discussed in Section 2, there is evidence in the literature that precise location hints have a positive effect on immediate performance, but it is not clear *a priori* if such effect would transfer to the context of formal specification. Also, there is some evidence that students struggle to understand Alloy counter-examples, but, as will be described below, we handpicked ideal counter-examples (e.g. minimal) and annotated them to make clear if they should be rejected or accepted by the correct specification.

3.5 Participants

Participants were recruited from the 2022/23 edition of a *Human-Computer Interaction* (HCI) course in the final semester of a *Computer Science* (CS) bachelor degree with a strong emphasis on *Software Engineering* (SE). To incentivize students to sign up, an extra 1 point in the final grade of the course (out of 20 possible) was offered (being an extra credit, students that did not sign up could still reach the maximum mark), and a total of 92 students volunteered to participate. As usual in most CS degrees, these students have attended courses on discrete maths and logic in the 1st year of the bachelor, where they were exposed to the basics of set theory and first-order logic. In the 2nd year of the bachelor they had some practice on the formal specification of simple imperative algorithms with pre- and post-conditions with first-order logic in an algorithmics course, and minimal exposure to OCL for UML models in a software development course, both relatively common subjects in CS and SE degrees. In both these courses they practised with pen-and-paper exercises, and this experiment would be, in principle, the first time they used a formal specification language with tool support for automatic analysis, and Alloy in particular. This background is similar to that of most CS and SE bachelors around the world, so we believe our sample is representative of the typical candidates to graduate level courses where formal specification is taught in depth (including most courses that teach Alloy).

Of the 92 students that signed up to participate (by signing the informed consent form in the welcome session), 2 were excluded

Figure 2: Alloy4Fun landing page with task #1 of the 1st session (treatment, i.e., hint message, omitted)

due to attrition (did not complete the procedure in the 2nd session), and 5 were excluded because they reported to have some prior knowledge of Alloy in the demographic survey. We ended up with 85 participants with the following demographics: 14 females, 70 males, 1 who preferred not to identify, with ages ranging from 19 to 29 years old (M = 20.8 years, SD = 1.5 years). The self-reported average grade (in a scale from 10 to 20) in previously completed courses in the bachelor degree ranged between 11.5 and 19.0 (M = 14.15, SD = 1.7), with one student preferring to not disclose it.

After the random assignment to the the 4 treatment groups and the exclusion of 7 participants, we ended up with the following distribution of the 85 students: N – 21, L – 25, E – 19, and D – 20. Since past grades could have a significant impact on the performance, we checked that the 4 groups were balanced in that respect. A Kruskal-Wallis test confirmed that the differences between the groups was not statistically relevant ($\chi^2(3) = 1.43$, p = 0.698).

3.6 Experimental objects

In the 1st session participants were asked to fix incorrect Alloy specifications of natural language requirements, while being supported by hints. In the 2nd session, participants were instead required to specify from scratch natural language requirements, without being supported by hints. The tasks of the 2nd session were more open since we wanted to measure the impact of hints in learning by performing more realistic posterior tasks.

All requirements in the tasks concerned 2 simple domain models, one of a photo sharing social network and another of a course management system. In each session half (6) of the incorrect specifications encoded requirements of the 1st domain model, and half (6) of the 2nd. Both the UML depiction of the models as class diagrams and their Alloy encoding were made available to the participants on the printed 1-page Alloy sheet, and the latter were also included in the respective Alloy4Fun tasks, as described in Section 3.7. These domain models were directly encoded in Alloy by declaring entities as signatures and relationships as fields (the encoding of the social network domain model in Alloy can be seen in the top of Figure 2).

Both the Alloy sheet and the Alloy fixing exercises of the 1st session focused on a small subset of the language that could reasonably be learned autonomously in the short duration of the experience. The tasks of the 2nd session could also be solved with this subset, which included: set operators union (`+`), intersection (`&`), and difference (`-`); relational dot-join composition (`.`); subset (`in`), equality (`=`), and inequality (`!=`) checks; cardinality checks (`no`, `some`, `lone`, `one`); Boolean connectives (`not`, `and`, `or`, `implies`), and universal quantification (`all`). The semantics of most of these operators was familiar to the participants (which had previous contact with first-order logic and set theory, as described in Section 3.5). The main exception was the relational composition operator, the most-used operator in the Alloy language and the key new concept participants had to learn during the experiment. We should stress that the dot-join composition operator of Alloy is different from the dot operator of object-oriented languages and OCL, namely always returning a set of related elements, and allowing fields to be navigated and chain-composed in any direction.

To (incorrectly) formalise the requirements we used the navigational style typical of Alloy, where specifications usually have very few quantifiers, and a combination of relational composition and set operators are used to compute sets of domain elements that should be somehow related or have a given cardinality. In the 1st session we made sure that students were exposed to all the above operators (except all the Boolean connectives, which they were well familiar with), ensuring that at least one incorrect specification used each operator. Relational composition was used in all the requirements. The incorrect specifications were always one error away from a correct one, and we designed the 12 tasks to cover the following 4 types of errors (3 tasks for each): one atomic *test* is incorrect; one set *operator* is incorrect; one relational *composition* is used incorrectly; or one (sub-)*expression* denoting a set is wrongly specified. Table 1 shows one task example for each type of error (for the social network domain model presented in Figure 2), where # denotes the position in which the task appears in the 1st session.

Our study focused on hints that could in theory be automatically generated and implemented in a platform such as Alloy4Fun. One of the goals of this study was to help us and the community decide which type of hint was worth further research and deployment. Since Alloy4Fun currently has no support for hints, in our experiment we manually curated the hint provided for each task. This is also one of the reasons for having the Alloy exercises of the 1st session designed as incorrect specifications to be fixed (for which we know what is the error), rather than having the participants write a specification from scratch – the other being that we wanted the first tasks to be completed in a reasonable time by practitioners with no Alloy background. We now briefly discuss the rational for including each type of hint, how they were hand-picked for each task, and how their generation could in principle be automated.

Error location (L). This is the most popular type of hint in environments for teaching programming. The erroneous syntactic element is highlighted, and for each type of error a standard sentence was formulated pointing to the highlighted element. For example, when the error is in a set operator, the sentence was *"Change the highlighted operator in the following incorrect specification"*. As seen in Section 2, this type of hint could be automatically generated by one of the existing Alloy repair techniques.

Counter-example (E). This is the usual feedback Alloy users get when they run analysis commands and thus it was natural to assess the impact of using visually depicted counter-examples. As

#	Error	Natural language requirement	Incorrect Alloy specification to be fixed
1	Test	Every image is posted by one user	`all p:Photo | lone posts.p`
2	Operator	Users cannot follow themselves	`all u:User | no u.follows-u`
4	Composition	Influencers are followed by everyone else	`all i:Influencer | i.follows = User-i`
5	Expression	Users that post an ad cannot post other kinds of photos	`(posts.Photo).posts in Ad`

Table 1: Task examples from the 1st session

mentioned above, these counter-examples can be automatically generated but there is some evidence that such (random) counter-examples are difficult to interpret by students. Thus, we hand-picked a minimal counter-example where the problem is clearly evident, and clarify in a sentence whether *"The example below is allowed by the following incorrect specification but should be forbidden"* or the other way around. Detecting to which of these two classes a counter-example belongs can be easily automated [26]. To generate minimal counter-examples, we could in principle use AlloyMax, an extension of Alloy [43] with support for minimisation / maximisation of relational expressions.

Error description (D). For this hint we included the sentence *"The following specification incorrectly states that…"* followed by a natural language sentence that describes the requirement that is wrongly being specified (instead of the desired one). This is a type of hint that we often give to students in our Alloy classes. With the recent advances in LLMs such hints could in principle be automatically generated, and thus we wanted to assess their impact in a controlled experiment, to help us decide if this was worth pursuing in future research.

3.7 Instrumentation

The experiment was instrumented in Alloy4Fun, a web-platform[3] for sharing, editing and analysing Alloy specifications, with an integrated counter-example visualiser. Compared to the Alloy Analyzer it introduces the notion of secret predicate, which allows the lecturers to define specification challenges where students must write semantically equivalent predicates. It has been used successfully in graduate formal specification courses for several years. Besides supporting autonomous studying, Alloy4Fun was also developed with the goal of supporting studies on formal methods education, and all interactions with the platform are anonymously recorded and available to researchers. The collected information includes the submitted specification, the parent specification (allowing the identification of sequential attempts), and the outcome of the analysis. Alloy4Fun datasets have already been used by researchers to evaluate Alloy repair techniques [4–6, 45]. Implementing this experiment required only minimal changes, namely implementing a timer, supporting persistent feedback messages and instances, and allowing the navigation through sequences of exercises.

The treatments for the different levels of the independent variable varied only in the shape of a persistent message presented below the editor in the 1st session (and an additional counter-example in the E level), as already described in Section 3.6. Figure 3 presents the hint feedback for task #1 presented in Table 1. All other UI elements remained unchanged between the different levels.

[3]http://alloy4fun.inesctec.pt/

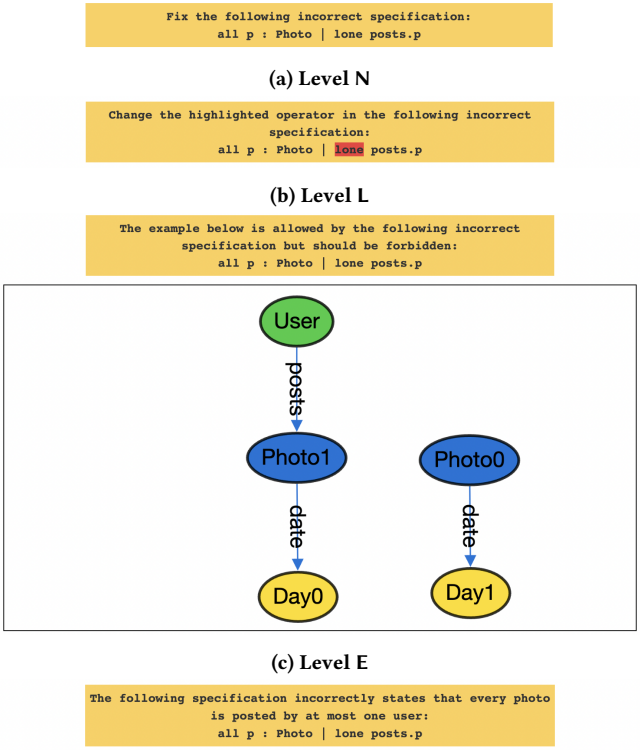


Figure 3: Treatment levels for task #1 of the 1st session

As shown in Figure 2, in each task of the 1st session participants are presented with an Alloy model encoding a domain model (signature declarations) and a predicate `spec` annotated with the expected requirement written in natural language pre-filled with an incorrect specification. The participants are expected to change the incorrect specification and then automatically check the correctness of their attempt using the button "Test Specification". At that point, they are informed that the attempt is correct (in which case they were blocked from further attempts and had to proceed to the next task using button "Next Challenge") or incorrect, without any further feedback. In the latter case, participants can perform additional attempts until the task times out. This interaction was similar in all treatment levels. The hint message remains the same during all attempts to solve one task (recall that these have been manually designed and not automatically generated). By editing the incorrect specification, participants lose access to the original incorrect specification to which the hint refers to, so all persistent messages also include the original incorrect specification (see Figure 3).

In the 2nd session, participants had a similar interface, except that rather than being pre-filled with an incorrect specification, the predicate they were expected to fill was empty. The persistent message simply stated "Write the missing specification above". Participants could still hit "Test Specification" multiple times and were informed whether the specification was correct or not.

To automatically check the correctness of the participants' submissions, we relied on the Alloy solver engine to test the equivalence of the participants' specifications against the oracle defined by us. This allows for specifications that are semantically equivalent to our oracle, but possibly syntactically different, to be still considered as correct. This technique has been used in Alloy courses to develop exercises in Alloy4Fun with support for auto-grading [26].

All the tasks, along with the curated hint feedback, were added to the Alloy4Fun database and made accessible through persistent links. For each participant, a unique sequence of 12 entries for each session was created in the database, one per task, and each participant ID was part of the persistent link to the 1st of those tasks. Having unique entries for each task and each participant was required in order to correctly identify the data of each participant (since Alloy4Fun is completely anonymous). Due to a technical error, for 3 of the 12 tasks in the E level, all of the 2nd domain model, the counter-example did not update correctly. We discuss the possible impact of this instrumentation error in Section 5.

This infrastructure allowed us to easily collect the 4 performance metrics. For each participant, we counted how many of the assigned 12 entries had a correct submission, thus obtaining PROD_1 and PROD_2; by additionally collecting all incorrect attempts to those entries, we calculated EFF_1 and EFF_2.

4 RESULTS

This section reports the results of the data analysis. For each operational variable we present the descriptive statistics and box-plots depicting data. For the inferential statistics, since several of our samples are not normally distributed we used the non-parametric Kruskal-Wallis test to check our null hypotheses. If a null hypothesis is rejected we used the *post-hoc* non-parametric Dunn's test with the Benjamini-Hochberg correction for multiple comparisons to determine pairwise significant differences.

4.1 Immediate performance

The descriptive statistics and box-plots for the immediate productivity PROD_1 are presented in Table 2a and Figure 4a, respectively. Group L, which had the hints with precise error locations, has a substantially higher mean and median, when compared to the others. A Kruskal-Wallis test confirms a significant difference among hint types ($\chi^2(3) = 32.356, p < 0.01$) and thus we accept the alternative hypothesis $H_1^{\mathsf{PROD}_1}$. The *post-hoc* Dunn's test only showed a significant difference among the L group and all the other groups (for the comparison L-N we have $z = -4.113, p < 0.01$, for L-E we have $z = -4.211, p < 0.01$ and for L-D we have $z = -5.024, p < 0.01$).

The descriptive statistics and box-plots for the immediate efficiency EFF_1 are presented in Table 2b and Figure 4b, respectively. Again, group L has a substantially higher mean and median, when compared to the others. A Kruskal-Wallis test confirms a significant difference among hint types ($\chi^2(3) = 24.706, p < 0.01$) and thus we accept the alternative hypothesis $H_1^{\mathsf{EFF}_1}$. Again, the *post-hoc* Dunn's test only showed a significant difference among the L group and all the other groups (for the comparison L-N we have $z = -3.567, p < 0.01$, for L-E we have $z = -3.586, p < 0.01$ and for L-D we have $z = -4.454, p < 0.01$).

Overall, the results suggest that, among the studied types of hints, only those with precise error locations can have a significant impact on the immediate performance.

4.2 Learning retention

The descriptive statistics and box-plots for the future productivity PROD_2 are presented in Table 2d and Figure 4d, respectively. The means and medians of all groups are quite similar, with group L (precise locations in 1st session) now having the lowest mean. The wide dispersion and range indicates a great amount of variability between participants. A Kruskal-Wallis test showed that the difference among hint types is not significant ($\chi^2(3) = 1.489, p = 0.68$) and thus we accept the null hypothesis $H_0^{\mathsf{PROD}_2}$.

The descriptive statistics and box-plots for the future efficiency EFF_2 are presented in Table 2e and Figure 4e, respectively. Again, the means and medians of all groups are now quite similar, with group L having the lowest mean and the control group N the highest. Likewise to the future productivity, a Kruskal-Wallis test showed that the difference among hint types is not significant ($\chi^2(3) = 1.315, p = 0.73$) and thus we accept the null hypothesis $H_0^{\mathsf{EFF}_2}$.

These results suggest that the three types of hints in our study have no significant impact on learning retention.

4.3 Emotional response

The descriptive statistics and box-plots for the immediate emotional response EMO_1 are presented in Table 2c and Figure 4c, respectively. A Kruskal-Wallis test confirms a significant difference among hint types ($\chi^2(3) = 12.358, p < 0.01$) and thus we accept the alternative hypothesis $H_1^{\mathsf{EMO}_1}$. The *post-hoc* Dunn's test only showed a strong significant difference among groups L and D ($z = -3.354, p < 0.01$) and borderline significance among L and N ($z = -2.386, p = 0.05$).

The descriptive statistics and box-plots for the future emotional response EMO_2 are presented in Table 2f and Figure 4f, respectively. We note that one student in group D did not submit the PrEmo 2 survey in the 2nd session, so this analysis considers only 19 subjects in that group. Unlike for the immediate emotional response, a Kruskal-Wallis test showed that in this case the difference among hint types is not significant ($\chi^2(3) = 3.713, p = 0.29$) and thus we accept the null hypothesis $H_0^{\mathsf{EMO}_2}$.

These results suggest that there is significant impact on the emotional response while learning with different types of hints, but that impact becomes insignificant once students start to solve future tasks without hints.

5 THREATS TO VALIDITY

Construct validity. To avoid mono-operation bias, we used two distinct domain models to define the tasks in our experiment. Our experiment may suffer from mono-method bias since each operational variable was defined in terms of a single measure. Concerning performance, measuring productivity with the number of accomplished tasks is common in user studies assessing the impact of

Statistic	N	L	E	D
Mean	5.67	9.12	5.42	4.75
Std. Err.	0.57	0.38	0.57	0.54
Std. Dev.	2.63	1.90	2.50	2.43
Median	6	9	5	4.5
Min	2	3	0	0
Max	11	12	11	9
Skewness	0.27	-1.46	0.11	-0.12
Kurtosis	-1.00	2.59	0.05	-0.92

(a) Statistics for $PROD_1$

Statistic	N	L	E	D
Mean	0.15	0.28	0.15	0.11
Std. Err.	0.03	0.03	0.03	0.02
Std. Dev.	0.15	0.13	0.14	0.10
Median	0.12	0.27	0.10	0.08
Min	0.02	0.09	0.00	0.00
Max	0.73	0.67	0.53	0.38
Skewness	2.51	0.89	1.51	1.29
Kurtosis	7.08	1.47	1.20	0.96

(b) Statistics for EFF_1

Statistic	N	L	E	D
Mean	-1.24	0.92	-1.37	-2.60
Std. Err.	0.93	0.60	0.80	0.72
Std. Dev.	4.28	3.00	3.47	3.23
Median	-1	1	-1	-1.5
Min	-8	-6	-10	-10
Max	11	7	4	1
Skewness	0.75	0.08	-0.80	-0.82
Kurtosis	1.04	0.10	0.08	-0.42

(c) Statistics for EMO_1

Statistic	N	L	E	D
Mean	5.95	4.84	5.89	4.95
Std. Err.	0.77	0.52	0.72	0.69
Std. Dev.	3.53	2.61	3.14	3.10
Median	5	5	5	5.5
Min	0	0	2	0
Max	12	11	12	11
Skewness	0.20	0.17	0.65	0.09
Kurtosis	-1.21	-0.37	-1.03	-1.25

(d) Statistics for $PROD_2$

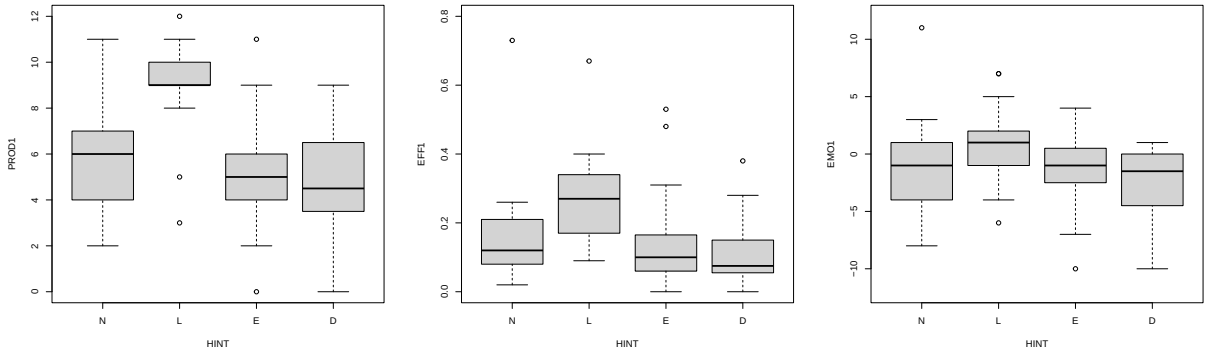
Statistic	N	L	E	D
Mean	0.21	0.13	0.15	0.15
Std. Err.	0.04	0.03	0.03	0.03
Std. Dev.	0.20	0.13	0.14	0.14
Median	0.11	0.90	0.11	0.11
Min	0.00	0.00	0.03	0.00
Max	0.67	0.47	0.50	0.44
Skewness	0.92	1.47	1.22	0.62
Kurtosis	-0.63	1.36	0.36	-1.04

(e) Statistics for EFF_2

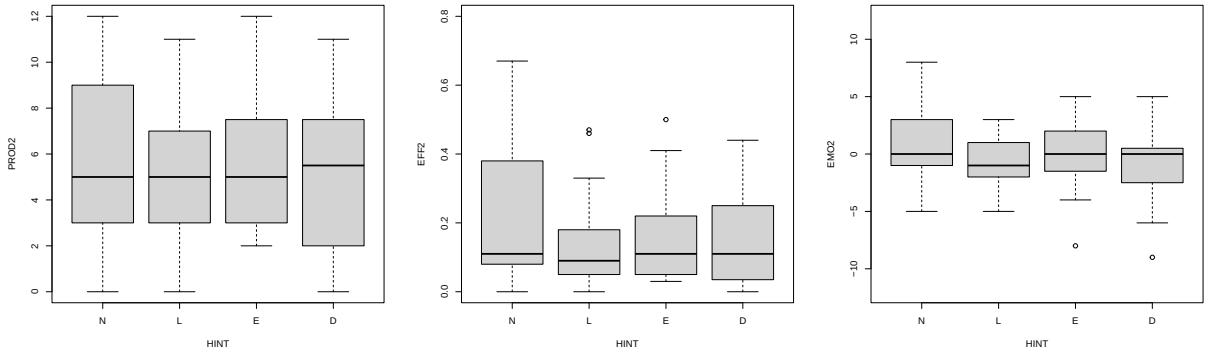
Statistic	N	L	E	D
Mean	0.90	-0.84	0.21	-0.68
Std. Err.	0.70	0.47	0.72	0.77
Std. Dev.	3.22	2.36	3.15	3.35
Median	0	-1	0	0
Min	-5	-5	-8	-9
Max	8	3	5	5
Skewness	0.41	-0.15	-0.61	-0.60
Kurtosis	-0.61	-1.17	0.32	0.14

(f) Statistics for EMO_2

Table 2: Descriptive statistics



(a) Box-plot for $PROD_1$ **(b) Box-plot for EFF_1** **(c) Box-plot for EMO_1**



(d) Box-plot for $PROD_2$ **(e) Box-plot for EFF_2** **(f) Box-plot for EMO_2**

Figure 4: Box-plots

hints. Some studies measure efficacy as the time to accomplish the tasks rather than using some metric involving the number of attempts. We chose the latter because we wanted to identify whether certain hint types were more prone to hint abuse, which would not be detected by measuring the time. Future performance in post-tests without hints has also been used in previous studies to measure learning retention. Although the used emotional response survey is standard, the aggregation of the emotions in a single metric is not standard in the literature but allowed us to perform a ballpark analysis of the emotional response. We checked if the results were identical with a more detailed analysis per emotion, and indeed for the immediate emotional response there was a significant difference for 5 of the emotions, while for future emotional response there was no significant difference for any of the emotions.

Internal validity. The participants performance may have been influenced by factors such as the clarity of the Alloy sheet, the formulation of the requirements in natural language, and the quality of the curated hints. Some of the authors of this work have been teaching Alloy for several years, and some of these exercises have in fact been developed for classes and refined over the years. Additionally, we also performed a very small pilot study with 2 researchers without Alloy background to further identify any possible issue.

Obviously, prior knowledge of Alloy could have had an impact on the results. To guarantee that all participants had the same Alloy background, we asked about prior Alloy knowledge in the demographic questionnaire and excluded from the analysis the participants that reported so. To avoid having the participants searching for external help during the experiment, proctors were assigned to all classrooms while the experiments were taking place.

As already stated, we had some instrumentation issues in the 1st session regarding the feedback of some tasks of the E group, all from the 2nd domain model. To check if this could have biased our analysis, we repeated the inferential statistics for the immediate performance taking into consideration only the 6 task relative to the 1st domain model, and all the conclusions were identical. We also computed the descriptive statistics separately for both domain models, and actually group E had a slight increase in productivity and efficacy in the tasks of the 2nd domain model, thus seeming to be unaffected by the error. It is worth noting that there was a slight decrease in the productivity for all the other groups.

Alloy4Fun is an academic tool and has not been properly validated for UX, which could in principle have affected the impact of hints. To check whether this was a factor, we assessed the results of the UMUX questionnaire, which produces an overall usability metric ranging from 0 to 100. While the overall values demonstrated only marginal usability (for the 1st session we have M = 53.4, SD = 19.4, and for the 2nd session we have M = 50.9, SD = 17.6, with non-acceptability being below 50 according to [39]), the difference in usability between the hint types was not statistically significant.

External validity. It is possible that the results of learning Alloy cannot be generalised to other formal specification languages. However, we have focused on a very small subset of operators that is common to most specification languages. The only operator that may not be standard in other languages is the composition that allows for the navigational style of Alloy. By focusing on a small subset of the language and fine-grained tasks, by design we are also not assessing the impact on learning how to use Alloy to specify full realistic models. To assess that a longitudinal study running during a full Alloy course would be more appropriate.

As already discussed in Section 3.5, the background of the participants is aligned with what is recommended by ACM for standard CS curricula, namely concerning the background on logic, discrete maths, and software engineering. As such we believe our conclusions can be generalised to most students majoring in CS.

Conclusion validity. Statistical relevance of the results was shown using non-parametric tests recommended for non-normal data.

6 DISCUSSION

We found it surprising that counter-examples (group E) and error descriptions (group D) did not have a positive impact on immediate performance. This may have due to cognitive overload of interpreting those hints, which may have lead to hint avoidance. Despite the instrumentation error in group E, the fact that this was the only group for which productivity increased during that session seems to corroborate this conjecture that participants may have started to disregard this type of hints after some tasks. Further evidence is that only a couple of students reported the error during the session. While we believe that counter-examples play an essential role in the validation of formal models, it is possible that their interpretation requires some training and thus they are not suitable for students in their first interactions with the language.

On the open-ended question in the 1st session questionnaire, participants expressed some thoughts about the provided hints. Some these are inline with the above analysis. In group D, students tended to find the help lacking, with comments such as "needs to be more explicit" or "information is not specific enough". In group E, responses were divided between finding it confusing and helpful. A student mentioned that it "helped me understand what I did wrong with the errors", and another stated that "At first, I didn't find it very clear, but over time, with practice, I understood it better".

Despite some initial concerns regarding their potential negative impact on learning retention, it seems that hints with precise error locations (group L) are beneficial. Immediate performance is better without a negative impact in learning retention, so students can acquire the same knowledge more efficiently. It is worth noting that no indication exists that participants in this group have brute-forced solutions, since their efficacy is higher than the other groups. Moreover, the emotional response was more positive while performing tasks with these hints, while a deprivation effect does not seem to have occurred after their withdrawal in the 2nd session. In the open-ended question, students from this group found the help useful and satisfactory, with comments like "The system's provided help that greatly assisted in quickly completing the tasks" or "I think the tips provided by the system were essential in helping to solve the problem, and without them, the solution would have been quite complicated".

Our results showed that only group L had a positive aggregated emotional response in the 1st session and that in the 2nd session there were no significant differences between the groups. As already reported, we also did a more detailed analysis per emotion, namely taking into consideration the difference between the self-reported Likert level after and before both sessions. The mean and standard

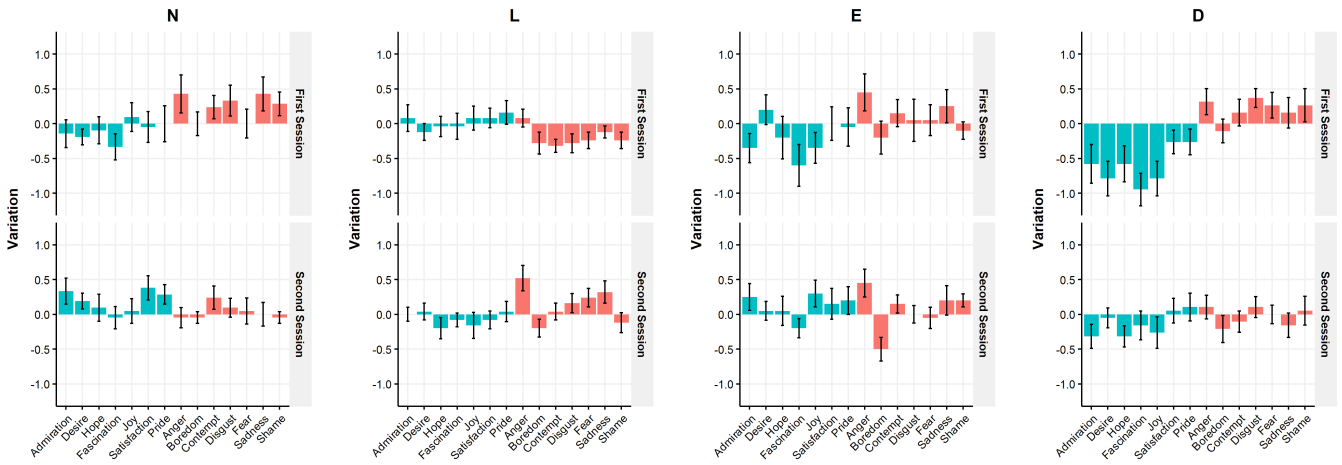


Figure 5: Emotional response after the 1st (top) and 2nd (bottom) sessions

error (the whiskers) of that variation are depicted in Figure 5. In the 1st session, and averaging all the groups there was a negative variation in all emotions with the exception of Boredom. The stronger decrease occurred in Fascination, Anger, Hope, and Joy. The Kruskal-Wallis test confirmed a significant difference between the groups for the following emotions: Desire, Fascination, Joy, Contempt, Shame, and Disgust. Looking at Figure 5 it seems that the hints in group D triggered the worst emotional response. A *post-hoc* analysis confirms that indeed the most significant difference is almost always between this group and group L.

With the exception of group L, the emotional response in the 2nd session improved for almost all of the different emotions, with an increase in the positive emotions (in blue) and a decrease in the negative ones (in red). This perhaps reflects some habituation with the language and the Alloy4Fun framework. Averaging all the groups, the stronger improvements were in Fascination, Desire, Joy, and Satisfaction, all emotions with a positive valence. In general, the best emotional response in this session was for group N. Even though there was an (expected) strong decrease in the emotional response of group L in the 2nd session – possibly caused by the withdrawal of hints – the effect was not sufficient to make the experience more frustrating than that of the other groups.

The open-ended question can shed some light on the emotional quantitative results. In all groups but L there were negative comments about the experience in the 1st session. In group N there were positive emotional responses ("I managed to answer the majority of the challenges correctly" or simply "I enjoyed the experience"), but also negative ("The system left me somewhat hesitant about the provided help"); in group E, students mentioned annoyance ("I was annoyed while doing this study") and frustration; and in group D, comments highlighted disappointment ("I am disappointed with my results in this test"), time pressure for responses, but also a decrease in boredom ("I am less bored than a moment ago"). In group L only one student mentioned feelings ("I felt frustrated not being able to finish some of the questions"). In general, responses in the 2nd session are also inline with the quantitative results. In group N only one student reported feeling frustrated and in D comments backed the improved response ("I feel that this week I learned more than last week, which makes me feel less frustrated" and "I am curious about the system"). In group E, despite the improved emotional response, participants still reported "It was a frustrating experience" or "I feel disappointed for not having done better". In group L comments were more mixed than in the 1st session, with a duality between "I managed to complete most of the challenges" and "I couldn't solve some of the exercises, so I'm disappointed and frustrated that I had to wait to move on to the next challenge".

7 CONCLUSION

This paper presented a study on the impact of different types of hints when learning a new formal specification language. As far as we know, it is the first such study focusing on learning formal specification rather than learning programming. The replication package for the experiment is publicly available[4]. The results showed that only hints highlighting error locations have a significant impact on immediate performance and emotional response, and none of the studied hints had a positive (or negative) impact on future performance (learning retention) and future emotional response. From this we can conclude that hints with precise error locations seem to be beneficial for the students, making the learning experience more pleasant and effective without apparent future negative effects. Our study focused on participants with no Alloy background and on fine-grained tasks, and some types of hints, namely counter-examples, may require a bit more experience to be useful. Further studies are required to assess the impact of hints over the full duration of a formal specification course, and also to assess the impact of other types of hints, namely hints combined with some sort of self-explanation prompts.

[4]To avoid compromising the anonymity of the review process, we've shared the replication package in the paper submission platform.

REFERENCES

[1] Vincent Aleven, Ido Roll, Bruce M. McLaren, and Kenneth R. Koedinger. 2016. Help Helps, But Only So Much: Research on Help Seeking with Intelligent Tutoring Systems. *Int. J. Artif. Intell. Educ.* 26, 1 (2016), 205–223.

[2] Elisa L. A. Baniassad, Lucas Zamprogno, Braxton Hall, and Reid Holmes. 2021. STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance. In *SIGCSE*. ACM, 1062–1068.

[3] G. H. Bower. 1992. How might emotions affect learning? In *The handbook of emotion and memory: Research and theory*, S.-Å. Christianson (Ed.). Lawrence Erlbaum Associates, Inc., 3–31.

[4] Simón Gutiérrez Brida, Germán Regis, Guolong Zheng, Hamid Bagheri, ThanhVu Nguyen, Nazareno Aguirre, and Marcelo F. Frias. 2021. Bounded Exhaustive Search of Alloy Specification Repairs. In *ICSE*. IEEE, 1135–1147.

[5] Simón Gutiérrez Brida, Germán Regis, Guolong Zheng, Hamid Bagheri, ThanhVu Nguyen, Nazareno Aguirre, and Marcelo F. Frias. 2022. ICEBAR: Feedback-Driven Iterative Repair of Alloy Specifications. In *ASE*. ACM, 55:1–55:13.

[6] Jorge Cerqueira, Alcino Cunha, and Nuno Macedo. 2022. Timely Specification Repair for Alloy 6. In *SEFM (LNCS, Vol. 13550)*. Springer, 288–303.

[7] Heeryung Choi, Jelena Jovanovic, Oleksandra Poquet, Christopher Brooks, Srecko Joksimovic, and Joseph Jay Williams. 2023. The benefit of reflection prompts for encouraging learning with hints in an online programming course. *Internet High. Educ.* 58 (2023), 100903.

[8] Robert Clarisó and Jordi Cabot. 2018. Fixing Defects in Integrity Constraints via Constraint Mutation. In *QUATIC*. IEEE Computer Society, 74–82.

[9] Albert T. Corbett and John R. Anderson. 2001. Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In *CHI*. ACM, 245–252.

[10] Antonio Damasio. 2005. *Descartes' error: Emotion, reason, and the human brain.* Penguin (Non-Classics), New York.

[11] Natasha Danas, Tim Nelson, Lane Harrison, Shriram Krishnamurthi, and Daniel J. Dougherty. 2017. User Studies of Principled Model Finder Output. In *SEFM (Lecture Notes in Computer Science, Vol. 10469)*. Springer, 168–184.

[12] Tristan Dyer, Tim Nelson, Kathi Fisler, and Shriram Krishnamurthi. 2022. Applying cognitive principles to model-finding output: The positive value of negative information. *Proc. ACM Program. Lang.* 6, OOPSLA1 (2022), 1–29.

[13] Kraig Finstad. 2010. The usability metric for user experience. *Interacting with computers* 22, 5 (2010), 323–327.

[14] A. Gupta, A. Elby, and B. A. Danielak. 2018. Exploring the entanglement of personal epistemologies and emotions in students' thinking. *Physical Review Physics Education Research* 14, 1 (2018), 010129.

[15] Luke Gusukuma, Austin Cory Bart, Dennis G. Kafura, and Jeremy Ernst. 2018. Misconception-Driven Feedback: Results from an Experimental Study. In *ICER*. ACM, 160–168.

[16] Regina Hebig, Truong Ho-Quang, Rodi Jolak, Jan Schröder, Humberto Linero, Magnus Ågren, and Salome Honest Maro. 2020. How do Students Experience and Judge Software Comprehension Techniques?. In *ICPC*. ACM, 425–435.

[17] Daniel Jackson. 2006. *Software abstractions: Logic, language, and analysis* (revised ed.). MIT Press.

[18] Daniel Jackson. 2019. Alloy: A language and tool for exploring software designs. *Commun. ACM* 62, 9 (2019), 66–76.

[19] Shriram Krishnamurthi and Tim Nelson. 2019. The Human in Formal Methods. In *FM (Lecture Notes in Computer Science, Vol. 11800)*. Springer, 3–10.

[20] Leslie Lamport. 2002. *Specifying Systems: The TLA+ language and tools for hardware and software engineers.* Addison-Wesley.

[21] Gaël Laurans and Pieter MA Desmet. 2017. Developing 14 animated characters for non-verbal self-report of categorical emotions. *Journal of Design Research* 15, 3-4 (2017), 214–233.

[22] Timotej Lazar, Aleksander Sadikov, and Ivan Bratko. 2018. Rewrite Rules for Debugging Student Programs in Programming Tutors. *IEEE Trans. Learn. Technol.* 11, 4 (2018), 429–440.

[23] Lu Li, Andrew Douglas Isherwood Gow, and Jiaxian Zhou. 2020. The Role of Positive Emotions in Education: A Neuroscience Perspective. *Mind, Brain, and Education* (2020). mbe.12244.

[24] Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. 2016. Lightweight specification and analysis of dynamic systems with rich configurations. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 373–383.

[25] Nuno Macedo, Alcino Cunha, and Tiago Guimarães. 2015. Exploring Scenario Exploration. In *FASE (Lecture Notes in Computer Science, Vol. 9033)*. Springer, 301–315.

[26] Nuno Macedo, Alcino Cunha, José Pereira, Renato Carvalho, Ricardo Silva, Ana C. R. Paiva, Miguel Sozinho Ramalho, and Daniel Castro Silva. 2021. Experiences on teaching Alloy with an automated assessment platform. *Sci. Comput. Program.* 211 (2021), 102690.

[27] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *SIGCSE (1)*. ACM, 931–937.

[28] Niloofar Mansoor, Hamid Bagheri, Eunsuk Kang, and Bonita Sharif. 2023. An Empirical Study Assessing Software Modeling in Alloy. In *FormaliSE*. IEEE, 44–54.

[29] Samiha Marwan, Nicholas Lytle, Joseph Jay Williams, and Thomas W. Price. 2019. The Impact of Adding Textual Explanations to Next-step Hints in a Novice Programming Environment. In *ITiCSE*. ACM, 520–526.

[30] Samiha Marwan and Thomas W. Price. 2023. iSnap: Evolution and Evaluation of a Data-Driven Hint System for Block-Based Programming. *IEEE Trans. Learn. Technol.* 16, 3 (2023), 399–413.

[31] Samiha Marwan, Joseph Jay Williams, and Thomas W. Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *ICER*. ACM, 61–70.

[32] Jessica McBroom, Irena Koprinska, and Kalina Yacef. 2022. A Survey of Automated Programming Hint Generation: The HINTS Framework. *ACM Comput. Surv.* 54, 8 (2022), 172:1–172:27.

[33] Joydeep Mitra. 2023. Studying the Impact of Auto-Graders Giving Immediate Feedback in Programming Assignments. In *SIGCSE (1)*. ACM, 388–394.

[34] Tim Nelson, Salman Saghafi, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. 2013. Aluminum: Principled scenario exploration through minimality. In *ICSE*. IEEE Computer Society, 232–241.

[35] R. Pekrun, T. Goetz, W. Titz, and R. P. Perry. 2002. Academic emotions in students' self-regulated learning and achievement: a program of qualitative and quantitative research. *Educational Psychology* 37 (2002), 91–105.

[36] Thomas W. Price, Samiha Marwan, Michael Winters, and Joseph Jay Williams. 2020. An Evaluation of Data-Driven Programming Hints in a Classroom Setting. In *AIED (2) (Lecture Notes in Computer Science, Vol. 12164)*. Springer, 246–251.

[37] Ruan Reis, Gustavo Soares, Melina Mongiovi, and Wilkerson de L. Andrade. 2019. Evaluating Feedback Tools in Introductory Programming Classes. In *FIE*. IEEE, 1–7.

[38] Kelly Rivers. 2017. *Automated Data-Driven Hint Generation for Learning Programming.* Ph. D. Dissertation. Carnegie Mellon University, USA.

[39] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research.* Morgan Kaufmann.

[40] P. Schutz and R. Pekrun (Eds.). 2007. *Emotion in education.* Academic Press, San Diego, CA.

[41] Kaiyuan Wang, Allison Sullivan, and Sarfraz Khurshid. 2018. Automated model repair for Alloy. In *ASE*. ACM, 577–588.

[42] Wengran Wang, Yudong Rao, Rui Zhi, Samiha Marwan, Ge Gao, and Thomas W. Price. 2020. Step Tutor: Supporting Students through Step-by-Step Example-Based Feedback. In *ITiCSE*. ACM, 391–397.

[43] Changjian Zhang, Ryan Wagner, Pedro Orvalho, David Garlan, Vasco Manquinho, Ruben Martins, and Eunsuk Kang. 2021. AlloyMax: bringing maximum satisfaction to relational specifications. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 155–167.

[44] Changjian Zhang, Ryan Wagner, Pedro Orvalho, David Garlan, Vasco M. Manquinho, Ruben Martins, and Eunsuk Kang. 2021. AlloyMax: Bringing maximum satisfaction to relational specifications. In *ESEC/SIGSOFT FSE*. ACM, 155–167.

[45] Guolong Zheng, ThanhVu Nguyen, Simón Gutiérrez Brida, Germán Regis, Nazareno Aguirre, Marcelo F. Frias, and Hamid Bagheri. 2022. ATR: template-based repair for Alloy specifications. In *ISSTA*. ACM, 666–677.