Wrong Answers for Wrong Reasons: The Risks of Ad Hoc Instruments

Luca Chiodini
luca.chiodini@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

Matthias Hauswirth
matthias.hauswirth@usi.ch
Software Institute, Università della Svizzera italiana
Lugano, Switzerland

ABSTRACT

To evaluate novel pedagogies, approaches, and tools, Computer Science Education researchers often conduct experiments to look for differences among groups treated with different interventions. The methodological rigor of such experiments affects the soundness of the conclusions the researchers can draw. In this paper we focus on a central aspect of such experimental research: the instruments used to assess participants' knowledge. Specifically, we study the use of ad hoc instruments and the risks due to their insufficient validation. We present a literature survey that highlights how, even though standardized instruments exist, the majority of published experiments in the last five years at major Computer Science Education conferences carries out pre/post-tests using ad hoc instruments, often with multiple-choice as question type. We demonstrate the risks of such commonly used but insufficiently validated multiple-choice instruments. We propose a richer way to analyze and assess the correctness of answers to multiple-choice questions, requiring participants to add brief explanation texts as a justification of each answer. We run an experiment and analyze the collected answers using the two approaches, with and without explanations, to show that the risk of drawing opposite conclusions from the statistical analysis is real.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **Student assessment**.

KEYWORDS

computing education; experiment; assessment; multiple-choice

ACM Reference Format:
Luca Chiodini and Matthias Hauswirth. 2021. Wrong Answers for Wrong Reasons: The Risks of Ad Hoc Instruments. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488042.3488045

Koli Calling '21, November 18–21, 2021, Joensuu, Finland

ACM ISBN 978-1-4503-8488-9/21/11…$15.00
https://doi.org/10.1145/3488042.3488045

1 INTRODUCTION

Since its very early days, Computer Science Education research has been concerned with providing empirical evidence to support innovations. In 2004, Fincher and Petre [14] highlighted *theory* and *evidence* as the two key dimensions that can be used to characterize research published in the area. They posited that good science should be rigorously backed both by appropriate theories and solid evidence, and that the field was still relatively immature. After more than ten years, Al-Zubidy et al. [1] performed an analysis that showed an increased adoption of methods that include empirical validation, observing an improvement: up to three times more papers provided such evidence, compared to previous findings [47].

Concerns about methodological rigor in Computer Science Education are still far from being over, though. Lishinski et al. [23] analyzed research published in a leading journal and a leading conference and discovered that newly published research seems to rely more often on theories, and that the "focus on empirical results in conference proceedings articles has surpassed that of journal publications". Despite this, they find that overall the methodological quality shows limited improvement.

The primary way in which novel pedagogies, approaches, and tools are evaluated beyond anecdotal reports is through experiments carried out with learners. Researchers design experiments in various ways, depending on which research questions they are trying to reason about and the practical constraints that are imposed by the real world. Those include, as examples, the number of students and classrooms reachable with a reasonable effort by the researcher, ethical concerns about putting a fraction of the studied population at a disadvantage when one other intervention is known to be likely effective, or the minimization of the disruption imposed by running the study in a regular university course or school class.

1.1 Standardized vs. ad hoc instruments

One of the key elements required to sustain rigorous experiments in education is the instrument used to assess participants' knowledge on one or more specific topics. Measurement validity (i.e., ensuring that we are measuring what we think we are measuring) and reliability (i.e., the instrument yields consistent measures) are important aspects to consider when choosing or designing an instrument. Margulieux [26] highlights that "how education researchers chose and implement measurements can have a meaningful impact on the validity and reliability of their findings". She calls for more standardization of instruments, because that would both "affords comparisons among studies" and "improves measurement tools by assessing their reliability and validity" [26]. In the rest of this paper, we adopt the term *standardized instrument* in this latter

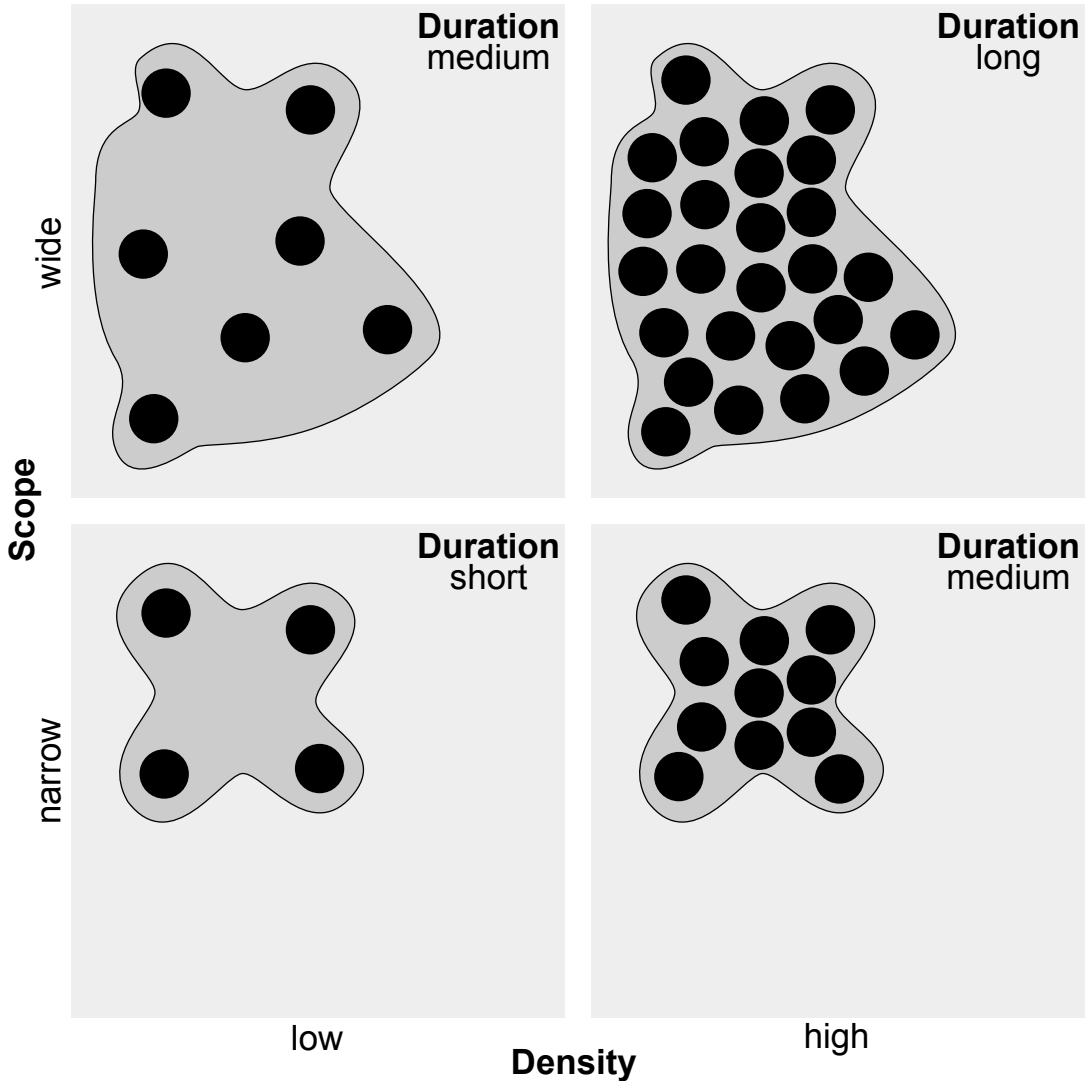

Figure 1: Trade-off between instrument scope and density.

meaning. Barkmin and Brinda [4] also stress the importance of using objective, reliable and validated instruments (i.e., standardized instruments). Worryingly, while reviewing programming assessments, they found that 10 out of 26 did not discuss reliability and that only 5 explicitly addressed validity.

One category of such standardized instruments consists of concept inventories, a prime way to assess students' knowledge, skills, and beliefs. Concept inventories contain questions in the form of multiple-choice items, address common misconceptions, constitute a reliable and validated instrument, and thus can also be administered as a post-test in experiments [2].

In the last years, the lack of concept inventories for Computer Science has been addressed by various pieces of work, ranging from relatively broad assessments to slightly more specific ones. Prominent examples include the SCS1 [33] for a general CS1 inventory, and BDSI for one targeted at basic data structures [35]. We believe that, even though public concept inventories and other standardized instruments exist, researchers still resort to developing their own custom instruments, which we call *ad hoc instruments* (in contrast to the standardized ones), because finding a suitable one is in general not so easy.

1.2 Scope and density of instruments

When selecting or designing instruments researchers have to deal with the trade-off between their *scope* and their *density* (Figure 1). The scope of an instrument represents the extent of the area it covers (e.g., all imperative programming skills, vs. skills related to variables). Its density determines how well its questions cover their scope with questions: low-density instruments cover the scope with a small number of questions while high-density instruments use many questions to cover most aspects within the scope.

Together, the scope and density of the instrument determine its duration (e.g., the time it takes to administer it, which is often correlated to the number of questions). Short instruments may neither cover a meaningful scope nor cover their scope with a useful density. Long instruments may cover a wide scope at a high density, but as a result they need to include a large number of questions and thus may have a long duration (as well as a high development cost). The more realistic instruments either have a narrow scope and a high density, or a wide scope and a low density. Some concept inventories fall into the wide scope and low density area: they cover a reasonably wide scope (so as to be more generally useful), but they have to limit the number of questions (so as to not take an excessive amount of time). To conduct an experiment with a narrow focus, researchers look for instruments with a narrow scope but a high density, and when they cannot find such instruments, create their own ad hoc instrument instead.

In this paper we focus on one of the most prevalent types of instrument: collections of multiple-choice questions (MCQs). Not only are MCQs easy to administer and easy to grade (potentially even automatically), but they also are heavily used in standardized instruments such as concept inventories [2]. We bring the following contributions:

(1) a literature survey that demonstrates the prevalence of the multiple-choice question type in experiments published at major Computer Science Education venues (Section 2),
(2) a discussion of a way to analyze answers to such questions that takes into account explanation texts (Section 3), and
(3) experimental evidence that ignoring explanations carries the concrete risk of undermining conclusions based on statistical tests (Section 4 and 5).

2 EXPERIMENTS AND INSTRUMENTS

2.1 Literature survey

To understand which kinds of experiments and assessments instruments researchers use to evaluate innovations in Computer Science Education, we conducted a literature survey. Inclusion criteria were research papers published at ICER, ITiCSE and Koli Calling (three main conferences whose proceedings are published on the ACM Digital Library) in the past five years, from 2016 to 2020 (inclusive), containing the word "posttest" or "post-test" in the full-text.

The initial set contained 64 papers in total. We then filtered the results to exclude papers spuriously included for a variety of wrong reasons, among which those whose core is (i) a literature review, (ii) the development of a concept inventory, (iii) just based on observations, (iv) not focused on content knowledge, or that more broadly do not describe an experiment. This selection yielded 31 papers broken down as shown in Table 1.

We made our best effort to characterize the main experiment described in those papers. Some of them, in addition to the knowledge pertaining to one or more Computer Science topics, report on other related aspects, such as the user evaluation of a tool, motivation, and intention to persist. The analysis presented here just focuses on the content knowledge, both in terms of the experiment and the assessment.

Table 1: Number of papers per conference (and in total) included in the literature survey.

Conference	Initial search	After filtering
ICER	18	11
ITiCSE	34	15
Koli Calling	12	5
Total	64	31

We captured in Table 2 two dimensions: the design of the experiment and the type of instrument used to assess students' understanding. On the experimental design aspect, we observe that only 7 out of 31 papers (23%) use a post-test only methodology, and that only 4 (13%) are not controlling the treatment condition with at least another condition. On the instrument used for the assessment, we focus on which *kind* of instrument has been used, classifying it into three categories:

- CI, when the instrument is a published concept inventory;
- Course, when the performance in various assessments normally administered during the course (e.g., a midterm exam, a final exam, or graded assignments) is recorded and used as instrument;
- Ad hoc, in all the other cases in which an instrument has been devised specifically for the purpose of the study[1]. Ad hoc instruments can have different question types: multiple-choice (abbreviated with MC in the table), short answers that might possibly be automatically graded, Parsons or coding problems, or custom exercises.

While concept inventories are considered the preferred way to conduct an assessment, because their validity and reliability have been previously evaluated, results from Table 2 indicate a very different reality. Out of 31 experiments, just 2 (6%) use a published concept inventory as instrument: Nelson et al. [32] use SCS1 [33], and Hinckle et al. [18] use MG-CSCI [36]. The remaining vast majority of papers (29 out of 31, 94%) use either assessments that are normally part of a course or that have been prepared ad hoc for the experiment.

We looked for explanations of this phenomenon in the papers which use an ad hoc instrument, but we could not find explicit arguments supporting the choice. Some of the reasons might trace back to a general lack of validated assessments in Computer Science due to some unique characteristics of the discipline [52], as well as a specific lack of concept inventories that go beyond the standard CS1 course, only partially addressed in recent years (e.g., with the BSDI concept inventory for basic data structures [35]).

Another reason might be attributed to the historical reluctance to publicly share instruments developed ad hoc for specific experiments. Out of 26 papers in that category, only 4 (15%) released the integral instrument. We acknowledge that concerns about instrument re-usability may affect the decision of the release, but we also clearly note that the current standard hinders the progress of the

[1]When researchers use course assessments just as an additional proxy metric to further evaluate students' performance, we still attribute the instrument used to the ad hoc category.

Table 2: Literature survey of papers that describe experiments in the past five years at three CSE conferences.

		Experimental design			Instrument	
Venue	Authors	Random[2]	Conditions[3]	Tests	Released[4]	Kind
ICER '20	Marwan et al. [27]	Yes	2	Pre+Post	No	Ad hoc (MC + programming)
ICER '19	DesPortes and DiSalvo [9]	Yes	2	Pre+Post	No	Ad hoc (draw circuits + MC)
ICER '19	Zhi et al. [54]	No	2	Post only	No	Course (assignments + project)
ICER '19	Marwan et al. [28]	Yes	3	Only post	No	Ad hoc (programming)
ICER '18	Ericson et al. [11]	Yes	4	Pre+Post	No	Ad hoc (MC + fix code + Parsons + write)
ICER '18	Gusukuma et al. [16]	No	2	Only post	No	Ad hoc (MC + programming)
ICER '17	Miljanovic and Bradbury [29]	/	1	Pre+Post	No	Ad hoc (MC)
ICER '17	Morrison [30]	Yes	3	Only post	No	Ad hoc (MC and open questions)
ICER '17	Margulieux and Catrambone [25]	Yes	6	Pre+Post	No	Ad hoc (MC + AppInventor problems)
ICER '17	Nelson et al. [32]	Yes	6	Pre+Post	/	CI (SCS1)
ICER '16	Morrison et al. [31]	Yes	2	Pre+Post	No	Ad hoc (MC)
ITiCSE '20	Kennedy et al. [21]	No	4	Pre+Post	No	Ad hoc (MC)
ITiCSE '20	Smith and Rixner [42]	Yes	2	Only post	No	Ad hoc (determine complexity)
ITiCSE '20	Anane and Alshammari [3]	Yes	2	Pre+Post	No	Ad hoc (MC)
ITiCSE '20	Domínguez et al. [10]	No	3	Pre+Post	No	Course (MC + exams)
ITiCSE '20	Hinckle et al. [18]	/	1	Pre+Post	/	CI (MG-CSCI)
ITiCSE '19	Soosai et al. [44]	No	2	Pre+Post	Yes	Ad hoc (open questions)
ITiCSE '19	Fessard et al. [13]	No	2	Pre+Post	No	Ad hoc (MC + programming)
ITiCSE '19	Walker et al. [49]	/	1	Pre+Post	No	Course (MC in midterm)
ITiCSE '18	Stephens-Martinez and Fox [45]	Yes	3	Only post	No	Ad hoc (short answer)
ITiCSE '18	Soltanpoor et al. [43]	No	2	Pre+Post	No	Ad hoc (short answer)
ITiCSE '17	Haidry et al. [17]	Yes	3	Pre+Post	No	Ad hoc (extract requirements)
ITiCSE '17	Cao and Porter [5]	Yes	2	Pre+Post	No	Ad hoc (MC/short answer)
ITiCSE '17	Scott and Ghinea [40]	Yes	2	Pre+Post	No	Ad hoc (MC)
ITiCSE '17	Wang et al. [50]	No	2	Pre+Post	Yes	Ad hoc (MC/short answer)
ITiCSE '16	Kumar [22]	Yes	2	Pre+Post	No	Ad hoc (exercises using the tool)
Koli Calling '20	Zavgorodniaia et al. [53]	Yes	3	Only post	No	Ad hoc (trace + MC)
Koli Calling '20	Tsarava et al. [46]	No	2	Pre+Post	Yes	Ad hoc (MC)
Koli Calling '19	Raj et al. [37]	No	2	Pre+Post	Yes	Ad hoc (short answer)
Koli Calling '18	Hosseini et al. [19]	/	1	Pre+Post	No	Ad hoc (write + Parsons)
Koli Calling '17	Ericson et al. [12]	Yes	3	Pre+Post	No	Ad hoc (MC + fix + Parsons + write)

research field. When researchers share their instruments, others may suggest improvements, produce refined versions, assess the validity and reliability, and reproduce results. Lastly, and possibly more importantly, experiments target a very specific and narrow set of concepts and skills, and they need to be backed by equally focused instruments (narrow scope, high density). Concept inventories are often pseudocode-based (e.g., SCS1 [33] and BDSI [35]). This can broaden their applicability, but at the same time it can reduce their usefulness in contexts that rely on a specific programming language. This also applies to the experiment reported in Section 4 that targets expressions, a concept that is valuable in a lot of programming languages and paradigms, but that depends on the specific knowledge of a specific programming language to determine whether a certain piece of source code is syntactically valid or to determine the semantic meaning of a construct. Existing concept inventories for programming tend instead to be wide in scope and low in density, conflicting with requirements.

These remarks are also reported in the literature. Decker and McGill hypothesize in their review [7] that the reason for the low adoption of standardized instruments may be the "lack of knowledge of the value of using an existing instrument" or "the challenge of finding such instruments".

For these reasons, efforts to collect, organize and present information about instruments in Computer Science are particularly valuable. We highlight two important collections: csedresearch.org,

[2]Whether participants have been randomly (at an individual granularity) allocated to one of the conditions.
[3]A condition may comprise multiple "groups" that, nonetheless, undergo the same treatment.
[4]Whether the instrument has been fully released, as a separate resource or described in its entirety within the paper.

a repository of computing education research instruments, and the list of programming assessments curated by Barkmin and Brinda [4]. They were built for different reasons and thus have a different focus. The former, at the time of this writing, lists 133 instruments in the "Computing" focus area, but only 6 instruments related to programming (i.e., classified as "CS 1 Concepts", "Programming", "Conditional Logic", "Functions", "Loops", or "Variables"). The latter contains instead only "programming assessments", classified using Barkmin's competency model.

2.2 Validity of ad hoc instruments

Table 2 shows the prevalence of ad hoc instruments in published experiments. When administering this kind of instruments, unlike a concept inventory whose validity has already been established, researchers should take particular care to get results that can be trusted. The extent to which the validity of such instruments is discussed varies greatly. We could find the following (non mutually exclusive) different approaches. Instruments have been:

- based on previously published assessments (e.g., [27] and [11]);
- tested in a pilot study with a smaller sample of students (e.g., [12] and [40]);
- "validated" by knowledgeable people in the domain (e.g., computer security experts in [3] and TAs in [40]);
- administered with a concurrent think-aloud with a subset of the students (e.g., [9]).

Despite the importance of the matter, in the majority of the papers analyzed in the survey researchers do not question the validity of the instrument they are using. In combination with the widely used multiple-choice type for questions, this seemingly innocuous approach can undermine experimental results. We proceed by proposing a richer way to analyze MCQs, and then showing in Section 4 a proof of the risk of not doing it using data from a real experiment we conducted.

3 ANALYZING MULTIPLE-CHOICE QUESTIONS

MCQs are often used in teaching and research. In teaching, to assess students in courses like CS1 [24], where the number of enrolled students is constantly increasing, the possibility to automatically grade the large number of answers is appealing. In quantitative research, to "have sufficient statistical power to detect at least medium-sized effects and allow for attrition" [38], it is important to recruit a large number of participants. However, bigger numbers imply significant costs, in terms of time and/or money, if the scoring of the questions cannot be automated. For this reason, MCQs are an attractive option.

MCQs are also believed to constitute an objective way of assessing knowledge [24], provided that enough care is put into the preparation of the items. Woodford and Bancroft [51] provide suggestions for writing effective MCQs.

In this paper we limit our discussion to the most common type of MCQs, in which only one option is meant to be selected as the correct answer. Multiple-answer MCQs have also been proposed [34] to overcome the limited solution space of single-answer ones, but they are less commonly used and fall outside the scope of this work.

3.1 Missing answers

To guide our reasoning throughout the remainder of this discussion, we will use a simple scheme to categorize answers to MCQs. Specifically, we classify each answer to a MCQ into one of the following three categories:

MC-Correct The answer matches the correct option (often referred to as the "key").
MC-Wrong The answer does not match the correct option.
MC-Missing No answer is provided.

We explicitly bring up the MC-Missing category, which is fundamentally different from the MC-Wrong one. Researchers seem to neglect this distinction: all the papers analyzed in our review did not explicitly comment on the presence and the treatment of such cases (with the sole exception of [40] in which is reported that "there were no cases with missing data").

Tests are mainly administered using two different formats: pen-and-paper and electronic. In the first case, instructors cannot easily control what students will do: there is the possibility of leaving one or more questions blank for a variety of reasons. Recurring reasons include: (i) students who are uncertain about the right answer and do not want to express an "educated guess"; (ii) students who completely lack the knowledge required to answer and do not want to express a "wild guess"; (iii) students who run out of time and leave behind some questions (not necessarily a contiguous block at the end of the test); and (iv) students who might simply not pay enough care and forget to answer some questions. Some of these scenarios also happen when questions are administered electronically, such as when the "mandatory answer" policy is not strictly enforced, or in the case of assessments that were not completed due to timing constraints.

It is important not to collapse MC-Missing into MC-Wrong because one might want to award different points for such answers (e.g., penalizing wrong answers but ignoring empty ones). Moreover, to allow reproducibility, researchers should always state how they deal with challenges (missing data points in this case).

3.2 Explain your reasoning

In the survey presented in Section 2, one paper stood out for taking a particular approach in combination with MCQs. Kennedy et al. [21] asked participants to "explain their reasoning", that is to write a brief sentence or paragraph on why they chose a particular option. Kennedy already employed this approach as one of the possible methodologies to elicit misconceptions from students [20].

We argue that pairing every MCQ with an explanation can provide invaluable insights into what is really going on inside students' minds, in ways that are not obtainable just from MCQs alone.

Focusing only on a free-text explanation, its assessment can be divided into four main categories:

Expl-Correct The explanation shows with enough strength that the student has the correct understanding required to solve the question.
Expl-Imprecise The explanation is clearly insufficient by itself to support a correct answer.
Expl-Wrong The explanation shows with enough strength that the student has the wrong understanding with respect to

the question posed and thus cannot properly answer the question.

Expl-Missing No (or nonsensical) explanation is provided.

When the multiple-choice answer is considered in conjunction with the explanation, the combinations between the answer provided to the MCQ and the explanation text generate $3 \times 4 = 12$ cases, which are summarized in Table 3, and deserve a detailed discussion.

Each cell in the table describes the overall category attributed to the answer as a whole (multiple-choice and explanation), essentially using the rich information provided in the explanation to "correct" or "triangulate" the answer given to the MCQ. We now discuss each row of Table 3 in order.

When students write a correct explanation (Expl-Correct), they have also probably selected the right option in the MCQ, and in that case the final judgment obviously treats the overall answer as correct (Correct). However, even with a correct explanation, in some cases the answer given to the MCQ is wrong. That is possibly due to contrived distractors or to students' inattention in selecting the option they really wanted to select, even though the explanation clearly shows that they have the proper understanding. We refer to the outcome in those cases using ProperExpl.

When students write a clearly imprecise explanation (Expl-Imprecise), one cannot assess with enough confidence their understanding, solely basing on the explanation. One thus has to rely upon the option selected in the MCQ.

When students write a wrong explanation (Expl-Wrong), they have also probably selected a wrong option in the MCQ, and the final judgment obviously treats the overall answer as wrong (Wrong). However, even with a wrong explanation, in some cases the answer given to the MCQ is correct. That is possibly due to a sequence of cancelling errors (Varney [48] recorded six cancelling errors from one student in an equation included in a MCQ), or students' inattention in selecting the option they really wanted to select, even though the explanation clearly shows that they did not perform the appropriate reasoning for reaching that conclusion. We call the outcome for this case BadExpl.

Finally, students might leave the explanation blank for a variety of reasons, including a shortage of time. While one cannot then gain insights about the reasoning made for wrong multiple-choice answers, it is worrying when it is not possible to determine whether the correct answer to the MCQ was given based on a proper understanding. We label NoExpl the case in which the answer to the MCQ was nonetheless correct.

In a perfect setup, only the combinations MC-Correct + Expl-Correct and MC-Wrong + Expl-Wrong would occur: students either select the right option after a proper chain of logically connected arguments that are then reported in written form in the explanation, or they select a wrong option due to some missing or wrong knowledge that is then reflected in the explanation. Unfortunately, this is hardly ever the case: administering assessments often results in non-ideal scenarios that still need to be managed.

We could not find literature references that explained how they dealt with such cases. Table 4 summarizes our best attempt, which we do not claim to be the only possible one, to reach a final decision on whether to award points to a certain answer, based on the combinations illustrated in Table 3.

We contrast it (first row of Table 4) with the usual approach in which students are just asked to answer to a MCQ without providing explanations.

When we believe the overall answer should be considered correct, a + is indicated. On the contrary, when an answer should be considered overall incorrect, we indicate a −. We limit ourselves to this binary classification for simplicity, considering missing answers as wrong. This is just one of the possible choices, as one might want to penalize wrong answers and treat neutrally the ones that are missing, or might want to discard entirely submissions that contain missing answers because, for instance, the researcher believes they would threaten the validity of an experiment. We urge researchers, no matter how they decide to proceed, to clearly state how they treat such cases in the reports on their experiments.

Looking at Table 4, there are four cases (highlighted in yellow) in which the final judgment on the correctness of an answer differs from the one attributed just by looking at the selected option in the multiple-choice item. In two cases, when the explanation is either wrong (Expl-Wrong) or missing (Expl-Missing), one (human or automated system) would judge the multiple-choice answer as correct and thus awards points, which is unfair because a student has not shown that they possess the required knowledge. In the other two cases, when the explanation would well support a correct reasoning (Expl-Correct), but the answer to the MCQ is wrong (MC-Wrong) or missing (MC-Missing), one would deem the multiple-choice answer incorrect and thus not award points, or award negative points, which is again unfair because the student has shown that they possess the required knowledge.

Obviously, these cases do not occur with very high frequency. However, their prevalence is probably bigger than what one would expect. As an example, we report in Section 4.3 the distribution of such cases in the pre-test of an experiment.

To illustrate with a concrete example sample answers that fall in the aforementioned categories, consider this MCQ that is part of the ad hoc instrument we developed for the experiment described in the next section. Given this Java class

```
public class Demo {
  public static void run() {
    int currentX = 1280;
    int baseX = 1000;
    System.out.println("Spaceship X-position: " + currentX - baseX);
  }
}
```

the question asks participants to select which one of the following options is true: (i) the compilation succeeds and an output message is printed on the screen after executing `Demo.run()` (ii) there is a compilation error (iii) there is a runtime error.

The correct answer is that a compilation error occurs: the compiler statically knows that the `+` operator will produce a value of type `String`, and that then the `-` operator cannot work when the first operand has type `String` and the second one `int`.

One student selected the first option in the MCQ (which would be classified as MC-Wrong), but wrote the following explanation that well supports the correct reasoning: "Since there is a String in the

Table 3: Categories for combinations of multiple-choice answers (columns) and explanations (rows).

	MC-Correct	MC-Wrong	MC-Missing
Expl-Correct	Correct	ProperExpl	ProperExpl
Expl-Imprecise	Correct	Wrong	Missing
Expl-Wrong	BadExpl	Wrong	Missing
Expl-Missing	NoExpl	Wrong	Missing

Table 4: Final judgment accounting for combinations of multiple-choice answers (columns) and explanations (rows).

	MC-Correct	MC-Wrong	MC-Missing
(multiple-choice only)	+	-	-
Expl-Correct	+	+	+
Expl-Imprecise	+	-	-
Expl-Wrong	-	-	-
Expl-Missing	-	-	-

print statement first, everything after it will be handled as a String. Therefore the only operator valid is '+' to concatenate Strings." With a rather high degree of confidence, we can classify this as Expl-Correct. The resulting combination is thus an instance of the ProperExpl category.

Conversely, the following are three instances that belong to the BadExpl category. All the three students selected the second option in the MCQ (classified as MC-Correct), but wrote wrong explanations: "you can't use - inside the println", "+currentX is invalid", and "you cannot compute operations inside a System.out.println()".

When MCQs are used in the context of a formative or summative assessment, the mistakes brought up in the previous paragraph might affect students' grades. When MCQs are instead used in a research instrument, these mistakes affect experimental results, to the extreme of flipping the findings resulting from statistical tests. We give a proof of this risk in the next section.

4 EXPERIMENT

We now illustrate the risks of relying on insufficiently validated MCQs instruments. We show this based on the example of a controlled experiment that uses a pre-/post-test to measure students' understanding of expressions. The goal of the experiment is to evaluate an intervention that presumably helps to teach expressions. The scope of this experiment, a certain subset of expressions in Java, is too narrow for standardized instruments. Thus, the development and use of an ad hoc instrument is necessary.

Expressions are considered an important topic of a Computer Science curriculum: after extensive Delphi processes, Goldman et al. [15] report on the high importance experts attribute to the two topics "construct/evaluate Boolean expressions" and "writing expressions for conditionals". Expressions however are not limited to arithmetic and conditions: a fragment of source code is normally full of expressions. Two small but illustrative examples in Java are "Method Invocation Expressions" and "Array Access Expressions"[5]. We believe that profoundly understanding expressions in terms of composing sub-expressions can greatly help students increase their conceptual knowledge.

We considered two different modalities (whose description is irrelevant for the scope of this paper) to augment the source code and highlight the nested structure of expressions. We will use Text to refer to the text-based modality, and Graphic to refer to the graphical one.

4.1 Experimental design

To verify whether one of the two modalities to teach expressions works better than the other one, we designed and conducted an experiment. We used a between-subjects experimental design, randomly assigning participants to two groups that differed only for the learning phase constituted by video-based explanations, worked examples and small exercises. The study procedure consisted of a pre-test, followed by the teaching intervention that used either the Text modality or the Graphic one to explain how expressions are constructed and evaluated in Java, and finally a post-test.

The pre-test consisted of 14 MCQs that asked students to predict what would happen after compiling one or more Java classes and executing `Demo.run()` (a static method always present that played the role of the "main" method). One of these questions[6] was presented at the end of Section 3.2. All of them had the same three options as possible answers (successful execution[7], compilation error, runtime error). We awarded one point for correct answers and zero points for wrong or missing answers. Each question is designed to target primarily, but not exclusively, a Java misconception. We considered for inclusion Java misconceptions from the progmiscon.org inventory [6] that are tagged with the "Expression" concept.

[5]The authoritative Java Language Specification dedicates a whole chapter to explain all the kinds of expressions present in the language.

[6]The full set of questions is available at https://zenodo.org/record/5118719/files/questions.md

[7]When students chose this first option, they had, in addition, to specify exactly what the output would be, and the answer was awarded the point only when also the output was correct.

The test was administered electronically through Moodle and was limited to 45 minutes. The same format with the same questions was also used as a post-test.

We recruited as participants students enrolled in a second-semester course focused on teaching Object-Oriented Programming with Java as a programming language. The experiment was conducted during one of the sessions normally used in the course. The participation in the study, which meant students' data collected and included in this research, was voluntary. Double points in the "participation" component of the final grade were awarded to those who consented to share their data for this research. The study was approved by the ethics committee.

We collected from participants who agreed to be part of the research study 44 pre-tests but only 40 valid post-tests, as some students did not complete the whole process or submitted an entirely empty post-test. Out of $N = 40$ participants, 21 were randomly assigned to the `Text` condition, and the remaining 19 students to the `Graphic` one.

The null hypothesis H_0 states that there is no difference in terms of average score between the two modalities. Conversely, H_a states that there is a difference in terms of average score between the two modalities. Given that N is not large and that we cannot assume a normal distribution, we use nonparametric statistical tests. We set $\alpha = 0.05$ as significance level. Results on the pre/post-test are reported as average ± standard deviation. As one point is awarded per correct question and there are no penalties, scores can range from 0 to 14.

4.2 Results using answers without explanations

The results reported in this section only depend on analyzing the answers given to the MCQs.

In the pre-test, participants in the `Text` condition scored an average of 9.81 ± 3.03 points. In the `Graphic` condition, the average number of points was 9.68 ± 2.11. A Mann-Whitney U test reported no statistically significant differences between the two conditions ($p = 0.3406$, $U = 184.0$).

As expected after the teaching intervention, scores in the post-test were higher for both conditions. Participants in the `Text` condition scored an average of 11.19 ± 2.75 points, while the ones in the `Graphic` condition obtained an average of 10.74 ± 2.23 points. A Mann-Whitney U test conducted on post-test scores still reported no statistically significant differences between the two conditions ($p = 0.1681$, $U = 164.0$). We therefore fail to reject H_0: we could not detect differences between the two conditions.

4.3 Results using answers revised with explanations

Following the approach described in Section 3.2, we required participants to write a textual explanation that justifies their selection (successful execution, compilation error, runtime error) in the MCQ. We reclassified all the answers, deciding to award a point or not based on the reasoning detailed in Table 4.

After an examination of the explanations, we excluded a student who was included in the Text condition but misunderstood the point of all the questions, probably due to a lack of background knowledge. In all the answers, the student claimed that the execution would have never succeeded as "one cannot invoke static methods like that". This reduced the size of the Text group from $N = 21$ to $N = 20$. Using the revised scores for all the participants, we re-analyzed the data at our disposal.

In the pre-test, participants in the `Text` condition scored an average of 9.15 ± 3.10 points. In the `Graphic` condition, the average number of points was 7.89 ± 3.16. A Mann-Whitney U test reported no statistically significant differences between the two conditions ($p = 0.1344$, $U = 150.5$).

Also in this scenario, as expected, scores in the post-test were higher for both conditions. Participants in the `Text` condition scored an average of 11.15 ± 2.39 points, while the ones in the `Graphic` condition obtained an average of 9.42 ± 3.11 points. A Mann-Whitney U test conducted on post-test scores now reported a statistically significant difference between the two conditions ($p = 0.0299$, $U = 123.0$). We therefore reject H_0 in favor of H_a. It appears that the `Text` modality is a better way to teach expressions, compared to the `Graphic` one.

5 REFLECTIONS ON THE RESULTS

Prior work in other fields already raised concerns about undisclosed choices in the way researchers collect and analyze data. Those choices can rather easily flip the outcome of an experiment, to the point of "allowing presenting anything as significant" [41]. In Section 4 we took that claim and went one step further. The use of MCQs in an ad hoc instrument carries the risk of drawing the wrong conclusion, despite efforts to carefully design the experiment.

We developed our instrument following a common approach used by other Computer Science Education researchers (Table 2): the first author created an initial set of MCQs to assess knowledge on expressions in Java. The choice of the questions was based already previously known misconceptions, following a recommendation to develop concept inventories [2]. The instrument was administered to other three experts, who gave feedback to improve the clarity of the questions, fix typos and select the most interesting ones that fit in the allocated time. This seemingly reasonable effort was evidently not enough to create an instrument suitable for experiments.

In Section 4.3 we brought up the case of the student who was focused on an entirely unrelated aspect (static methods) and was missing the point of the questions. That student, in the raw analysis described in Section 4.2, was improperly getting points for all questions that still had as a correct option "there is a compilation error", but for entirely unrelated reasons. The questions were not actually measuring the understanding of that student. Eliminating or even anticipating such issues is hard, as some of them are intrinsically related to the topics that are being tested.

Similar cases are scattered throughout the answers to the different questions. Figure 2 shows how many such cases were found after analyzing the explanations given in the pre-test (regardless of the condition in which participants were in, as the tests did not differ) and categorizing the answers according to Table 3. We note that there is a difference in prevalence across the 14 questions, with some noteworthy patterns emerging.

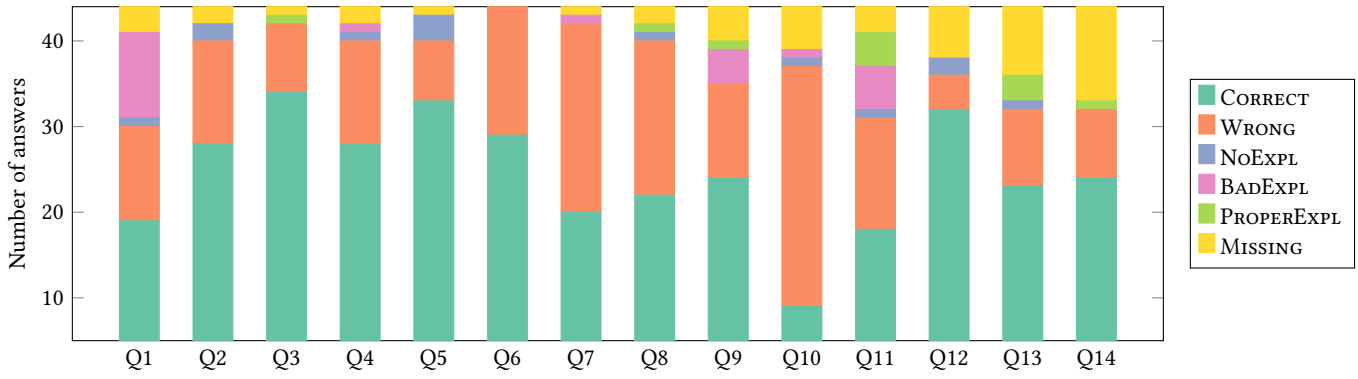

Figure 2: Distribution of assessment categories per question (pre-test, all groups, $N = 44$).

The prevalence of answers that would be correctly judged even without explanations (i.e., those belonging to the categories Correct and Wrong) greatly varies across questions. They constitute the totality in Question 6, but they account for just 68% in Question 1. The average is at 85%, which means that the remaining 15% is potentially wrongly judged if one does not consider explanations. Questions towards the end of the test (the order of questions was not shuffled) tend to exhibit higher rates of missing answers, possibly due to students' tiredness or lack of time.

While it can be claimed that these issues can be mitigated by increasing the number of participants in the experiments, our group sizes are common in the analyzed literature. We screened all the 31 papers included in the literature survey detailed in Table 2 to note what was the number of participants in the experimental condition that had fewer of them. We made an educated guess to deduce N where it was not explicitly specified (e.g., by looking at the degrees of freedom of the reported statistics). We found that $N \leq 30$ in the majority of the papers (16 out of 31). This is also in line with the finding reported in a systematic literature review that included many computing education venues by Decker et al. [8]: the most common value for N was in the 20-29 range.

6 LIMITATIONS

In our experiment, triangulating the multiple-choice answers with explanations flipped the conclusion one would draw from a statistical test. This does not necessarily generalize to *all* experiments, and would need to be corroborated. Moreover, only the first author was involved in the categorization of the explanations. Other researchers could adopt the same style of analysis in other experiments to see how much this finding applies to different contexts.

Data from the literature survey condensed in Table 2 has been collected by the first author, who made his best effort to categorize the kind of instrument used in the main experiment described in each work.

Limiting the literature survey to papers that contain the terms "posttest" or "post-test" may have excluded studies that do not use those very words to denote that idea. We still deem this choice appropriate as it targets works that contain experiments with some sort of evaluation after the learning intervention, and it also retains experiments that do not necessarily use a pre-test to measure knowledge before the experiment or that do not compare results across multiple conditions. While we believe that the resulting set constitutes a representative sample of experiments published in Computer Science Education venues, it should by no means be treated as an exhaustive review of all experiments published in all venues.

7 RECOMMENDATIONS FOR EXPERIMENTS

While it is easy to report statistics that prove or disprove the effectiveness of a certain innovation, it is problematic to trust the results when they are so susceptible to seemingly reasonable interpretations of the answers given to an instrument. Recruiting a large number of students with the appropriate prior knowledge can be challenging due to various external constraints. The issues described in this paper tend to affect more smaller-sized groups, such as those commonly reported in the analyzed papers. Increasing the groups' sizes can mitigate problems, such as decreasing the weight of students who completely misunderstand the meaning of all the questions, or those who run out of time and rush to complete in a non-principled way the answers towards the end of a test.

However, since we are well aware that such an increase is often unfeasible, we recommend two possible ways to address the problem and mitigate such issues, so that results can be trusted more. Researchers can evaluate which one suits best their needs, depending on the specific context of the experiment.

The first one is to *use a standardized instrument (e.g., a concept inventory)*. Whenever possible, researchers should rely on instruments that have been specifically developed and studied to be as reliable and valid as possible.

The alternative is to *add explanations to MCQs*. We described why sometimes using a concept inventory is unfeasible (e.g., an inventory to assess the specific topics under scrutiny does not exist) and recruiting more participants is challenging. In those cases, asking participants to always provide explanations in addition to selecting the option in MCQs and then judge the correctness of the answers taking into account those explanations can be an effective way to get more valid and reliable measures of participants' knowledge.

8 CONCLUSION

We introduced a way to categorize answers to MCQs, surveyed the literature on experiments in Computer Science Education research to prove the prevalence of MCQs in ad hoc instruments frequently used in experiments, and showed in a real experiment how ignoring explanations that justify participants' answers can lead to drawing wrong conclusions. We hope that this work can inspire researchers to take these aspects into account when designing instruments and conducting experiments in the future, to increase the rigor of the work published in the field. As an additional practice that we hope will become more broadly adopted by the community, we share at https://zenodo.org/record/5118719 the raw (anonymized) data that allow the replication of the reported statistics [39] about the experiment.

ACKNOWLEDGMENTS

This work was partially funded by the Swiss National Science Foundation project 200021_184689.

REFERENCES

[1] Ahmed Al-Zubidy, Jeffrey C. Carver, Sarah Heckman, and Mark Sherriff. 2016. A (Updated) Review of Empiricism at the SIGCSE Technical Symposium. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, Memphis Tennessee USA, 120–125. https://doi.org/10.1145/2839509.2844601

[2] Vicki L. Almstrum, Peter B. Henderson, Valerie Harvey, Cinda Heeren, William Marion, Charles Riedesel, Leen-Kiat Soh, and Allison Elliott Tew. 2006. Concept Inventories in Computer Science for the Topic Discrete Mathematics. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR '06)*. ACM, New York, NY, USA, 132–145. https://doi.org/10.1145/1189215.1189182

[3] Rachid Anane and Mohammad T. Alshammari. 2020. A Dynamic Visualisation of the DES Algorithm and a Multi-Faceted Evaluation of Its Educational Value. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 370–376. https://doi.org/10.1145/3341525.3387386

[4] Mike Barkmin and Torsten Brinda. 2020. Analysis of Programming Assessments — Building an Open Repository for Measuring Competencies. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–10. https://doi.org/10.1145/3428029.3428039

[5] Yingjun Cao and Leo Porter. 2017. Impact of Performance Level and Group Composition on Student Learning during Collaborative Exams. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Bologna Italy, 152–157. https://doi.org/10.1145/3059009.3059024

[6] Luca Chiodini, Igor Moreno Santos, Andrea Gallidabino, Anya Tafliovich, André L. Santos, and Matthias Hauswirth. 2021. A Curated Inventory of Programming Language Misconceptions. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 380–386. https://doi.org/10.1145/3430665.3456343

[7] Adrienne Decker and Monica M. McGill. 2019. A Topical Review of Evaluation Instruments for Computing Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis MN USA, 558–564. https://doi.org/10.1145/3287324.3287393

[8] Adrienne Decker, Monica M. McGill, and Amber Settle. 2016. Towards a Common Framework for Evaluating Computing Outreach Activities. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*. ACM Press, Memphis, Tennessee, USA, 627–632. https://doi.org/10.1145/2839509.2844567

[9] Kayla DesPortes and Betsy DiSalvo. 2019. Trials and Tribulations of Novices Working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, Toronto ON Canada, 219–227. https://doi.org/10.1145/3291279.3339427

[10] Adrián Domínguez, Luis de-Marcos, and José-Javier Martínez-Herráiz. 2020. Effects of Competitive and Cooperative Classroom Response Systems on Quiz Performance and Programming Skills in a Video Game Programming Course. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 398–403. https://doi.org/10.1145/3341525.3387393

[11] Barbara J. Ericson, James D. Foley, and Jochen Rick. 2018. Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM, Espoo Finland, 60–68. https://doi.org/10.1145/3230977.3231000

[12] Barbara J. Ericson, Lauren E. Margulieux, and Jochen Rick. 2017. Solving Parsons Problems versus Fixing and Writing Code. In *Proceedings of the 17th Koli Calling Conference on Computing Education Research - Koli Calling '17*. ACM Press, Koli, Finland, 20–29. https://doi.org/10.1145/3141880.3141895

[13] Grégoire Fessard, Patrick Wang, and Ilaria Renna. 2019. Are There Differences in Learning Gains When Programming a Tangible Object or a Simulation?. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Aberdeen Scotland Uk, 78–84. https://doi.org/10.1145/3304221.3319747

[14] Sally Fincher and Marian Petre (Eds.). 2005. *Computer Science Education Research*. Taylor & Francis. 95–110 pages. https://doi.org/10.1201/9781482287325-18

[15] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. 2008. Identifying Important and Difficult Concepts in Introductory Computing Courses Using a Delphi Process. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, New York, NY, USA, 256–260. https://doi.org/10.1145/1352135.1352226

[16] Luke Gusukuma, Austin Cory Bart, Dennis Kafura, and Jeremy Ernst. 2018. Misconception-Driven Feedback: Results from an Experimental Study. In *Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18*. ACM Press, Espoo, Finland, 160–168. https://doi.org/10.1145/3230977.3231002

[17] Shifa-e-Zehra Haidry, Katrina Falkner, and Claudia Szabo. 2017. Identifying Domain-Specific Cognitive Strategies for Software Engineering. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Bologna Italy, 206–211. https://doi.org/10.1145/3059009.3059032

[18] Madeline Hinckle, Arif Rachmatullah, Bradford Mott, Kristy Elizabeth Boyer, James Lester, and Eric Wiebe. 2020. The Relationship of Gender, Experiential, and Psychological Factors to Achievement in Computer Science. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 225–231. https://doi.org/10.1145/3341525.3387403

[19] Roya Hosseini, Kamil Akhuseyinoglu, Andrew Petersen, Christian D. Schunn, and Peter Brusilovsky. 2018. PCEX: Interactive Program Construction Examples for Learning Programming. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–9. https://doi.org/10.1145/3279720.3279726

[20] Cazembe Kennedy and Eileen T. Kraemer. 2018. What Are They Thinking?: Eliciting Student Reasoning About Troublesome Concepts in Introductory Computer Science. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–10. https://doi.org/10.1145/3279720.3279728

[21] Cazembe Kennedy, Aubrey Lawson, Yvon Feaster, and Eileen Kraemer. 2020. Misconception-Based Peer Feedback: A Pedagogical Technique for Reducing Misconceptions. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 166–172. https://doi.org/10.1145/3341525.3387392

[22] Amruth N. Kumar. 2016. The Effectiveness of Visualization for Learning Expression Evaluation: A Reproducibility Study. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Arequipa Peru, 192–197. https://doi.org/10.1145/2899415.2899427

[23] Alex Lishinski, Jon Good, Phil Sands, and Aman Yadav. 2016. Methodological Rigor and Theoretical Foundations of CS Education Research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, Melbourne VIC Australia, 161–169. https://doi.org/10.1145/2960310.2960328

[24] Raymond Lister. 2000. On Blooming First Year Programming, and Its Blooming Assessment. In *Proceedings of the Australasian Conference on Computing Education - ACSE '00*. ACM Press, Melbourne, Australia, 158–162. https://doi.org/10.1145/359369.359393

[25] Lauren Margulieux and Richard Catrambone. 2017. Using Learners' Self-Explanations of Subgoals to Guide Initial Problem Solving in App Inventor. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, Tacoma Washington USA, 21–29. https://doi.org/10.1145/3105726.3106168

[26] Lauren Margulieux, Tuba Ayer Ketenci, and Adrienne Decker. 2019. Review of Measurements Used in Computing Education Research and Suggestions for Increasing Standardization. *Computer Science Education* 29, 1 (Jan. 2019), 49–78. https://doi.org/10.1080/08993408.2018.1562145

[27] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. ACM, Virtual Event New Zealand, 194–203. https://doi.org/10.1145/3372782.3406264

[28] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, Toronto ON Canada, 61–70. https://doi.org/10.1145/3291279.3339420

[29] Michael A. Miljanovic and Jeremy S. Bradbury. 2017. RoboBUG: A Serious Game for Learning Debugging Techniques. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, Tacoma Washington USA, 93–100. https://doi.org/10.1145/3105726.3106173

[30] Briana B. Morrison. 2017. Dual Modality Code Explanations for Novices: Unexpected Results. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, Tacoma Washington USA, 226–235. https://doi.org/10.1145/3105726.3106191

[31] Briana B. Morrison, Adrienne Decker, and Lauren E. Margulieux. 2016. Learning Loops: A Replication Study Illuminates Impact of HS Courses. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, Melbourne VIC Australia, 221–230. https://doi.org/10.1145/2960310.2960330

[32] Greg L. Nelson, Benjamin Xie, and Andrew J. Ko. 2017. Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. In *Proceedings of the 2017 ACM Conference on International Computing Education Research - ICER '17*. ACM Press, Tacoma, Washington, USA, 2–11. https://doi.org/10.1145/3105726.3106178

[33] Miranda C. Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. Association for Computing Machinery, New York, NY, USA, 93–101. https://doi.org/10.1145/2960310.2960316

[34] Andrew Petersen, Michelle Craig, and Paul Denny. 2016. Employing Multiple-Answer Multiple Choice Questions. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Arequipa Peru, 252–253. https://doi.org/10.1145/2899415.2925503

[35] Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C. Webb, Cynthia Lee, and Michael Clancy. 2019. BDSI: A Validated Concept Inventory for Basic Data Structures. In *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER '19)*. ACM, New York, NY, USA, 111–119. https://doi.org/10.1145/3291279.3339404

[36] Arif Rachmatullah, Bita Akram, Danielle Boulden, Bradford Mott, Kristy Boyer, James Lester, and Eric Wiebe. 2020. Development and Validation of the Middle Grades Computer Science Concept Inventory (MG-CSCI) Assessment. *EURASIA Journal of Mathematics, Science and Technology Education* 16, 5 (Feb. 2020). https://doi.org/10.29333/ejmste/116600

[37] Adalbert Gerald Soosai Raj, Hanqi Zhang, Viren Abhyankar, Saswati Mukerjee, Eda Zhang, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2019. Impact of Bilingual CS Education on Student Learning and Engagement in a Data Structures Course. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–10. https://doi.org/10.1145/3364510.3364518

[38] Jeremy Roschelle, Mingyu Feng, H. Alix Gallagher, Robert Murphy, Christopher Harris, Danae Kamdar, and Gucci Trinidad. 2014. Recruiting Participants for Large-Scale Random Assignment Experiments in School Settings. *Grantee Submission* (2014).

[39] Kate Sanders, Judy Sheard, Brett A. Becker, Anna Eckerdal, Sally Hamouda, and Simon. 2019. Inferential Statistics in Computing Education Research: A Methodological Review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER '19)*. ACM, New York, NY, USA, 177–185. https://doi.org/10.1145/3291279.3339408

[40] Michael James Scott and Gheorghita Ghinea. 2017. On the Educational Impact of Lecture Recording Reduction: Evidence from a Randomised Trial. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Bologna Italy, 287–292. https://doi.org/10.1145/3059009.3059037

[41] Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. 2011. False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant. *Psychological Science* 22, 11 (Nov. 2011), 1359–1366. https://doi.org/10.1177/0956797611417632

[42] Rebecca Smith and Scott Rixner. 2020. Compigorithm: An Interactive Tool for Guided Practice of Complexity Analysis. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 363–369. https://doi.org/10.1145/3341525.3387390

[43] Reza Soltanpoor, Charles Thevathayan, and Daryl D'Souza. 2018. Adaptive Remediation for Novice Programmers through Personalized Prescriptive Quizzes. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Larnaca Cyprus, 51–56. https://doi.org/10.1145/3197091.3197097

[44] Adalbert Gerald Soosai Raj, Eda Zhang, Saswati Mukherjee, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2019. Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Aberdeen Scotland Uk, 499–505. https://doi.org/10.1145/3304221.3319787

[45] Kristin Stephens-Martinez and Armando Fox. 2018. Giving Hints Is Complicated: Understanding the Challenges of an Automated Hint System Based on Frequent Wrong Answers. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Larnaca Cyprus, 45–50. https://doi.org/10.1145/3197091.3197102

[46] Katerina Tsarava, Manuel Ninaus, Tereza Hannemann, Kristina Volná, Korbinian Moeller, and Cyril Brom. 2020. Fostering Knowledge of Computer Viruses among Children: The Effects of a Lesson with a Cartoon Series. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–9. https://doi.org/10.1145/3428029.3428033

[47] David W. Valentine. 2004. CS Educational Research: A Meta-Analysis of SIGCSE Technical Symposium Proceedings. *ACM SIGCSE Bulletin* 36, 1 (2004), 255–259.

[48] Robert N. Varney. 1998. More Remarks on Multiple Choice Questions. *American Journal of Physics* 52, 12 (June 1998), 1069. https://doi.org/10.1119/1.13762

[49] James Walker, Man Wang, Steven Carr, Jean Mayo, and Ching-Kuang Shene. 2019. Teaching Integer Security Using Simple Visualizations. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Aberdeen Scotland Uk, 513–519. https://doi.org/10.1145/3304221.3319760

[50] Man Wang, Jean Mayo, Ching-Kuang Shene, Steve Carr, and Chaoli Wang. 2017. UNIXvisual: A Visualization Tool for Teaching UNIX Permissions. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Bologna Italy, 194–199. https://doi.org/10.1145/3059009.3059031

[51] Karyn Woodford and Peter Bancroft. 2005. Multiple Choice Questions Not Considered Harmful. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42 (ACE '05)*. Australian Computer Society, Inc., AUS, 109–116.

[52] Aman Yadav, David Burkhart, Daniel Moix, Eric Snow, Padmaja Bandaru, and Lissa Clayborn. 2015. Sowing the Seeds: A Landscape Study on Assessment in Secondary Computer Science Education. (July 2015).

[53] Albina Zavgorodniaia, Arto Hellas, Otto Seppälä, and Juha Sorva. 2020. Should Explanations of Program Code Use Audio, Text, or Both? A Replication Study. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. ACM, Koli Finland, 1–10. https://doi.org/10.1145/3428029.3428050

[54] Rui Zhi, Min Chi, Tiffany Barnes, and Thomas W. Price. 2019. Evaluating the Effectiveness of Parsons Problems for Block-Based Programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, Toronto ON Canada, 51–59. https://doi.org/10.1145/3291279.3339419