Feedback on Student Programming Assignments: Teaching Assistants vs Automated Assessment Tool

Nynne Grauslund Kristiansen
CCER: Center for Computing Education Research
IT University of Copenhagen (ITU)
Denmark

Sebastian Mateos Nicolajsen
CCER: Center for Computing Education Research
IT University of Copenhagen (ITU)
Denmark

Claus Brabrand
CCER: Center for Computing Education Research
IT University of Copenhagen (ITU)
Denmark

ABSTRACT

Existing research does not quantify and compare the differences between automated and manual assessment in the context of feedback on programming assignments. This makes it hard to reason about the effects of adopting automated assessment at the expense of manual assessment. Based on a controlled experiment involving N=117 undergraduate first-semester CS1 students, we compare the effects of having access to feedback from: i) only automated assessment, ii) only manual assessment (in the form of teaching assistants), and iii) both automated as well as manual assessment. The three conditions are compared in terms of (objective) task effectiveness and from a (subjective) student perspective.

The experiment demonstrates that having access to both forms of assessment (automated *and* manual) is superior both from a task effectiveness as well as a student perspective. We also find that the two forms of assessment are complementary: automated assessment appears to be better in terms of task effectiveness; whereas manual assessment appears to be better from a student perspective. Further, we found that automated assessment appears to be working better for men than women, who are significantly more inclined towards manual assessment. We then perform a cost/benefit analysis which leads to the identification of four equilibria that appropriately balance costs and benefits. Finally, this gives rise to four recommendations of when to use which kind or combination of feedback (manual and/or automated), depending on the number of students and the amount of per-student resources available. These observations provide educators with evidence-based justification for budget requests and considerations on when to (not) use automated assessment.

CCS CONCEPTS

• **Social and professional topics → Student assessment**; **CS1**; **Computer science education**.

KEYWORDS

automated assessment, teaching assistants, student experiments, feedback

Koli Calling, November 16–19, 2023, Koli, Finland
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-8488-9/22/11…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Nynne Grauslund Kristiansen, Sebastian Mateos Nicolajsen, and Claus Brabrand. 2023. Feedback on Student Programming Assignments: Teaching Assistants vs Automated Assessment Tool. In *23rd Koli Calling International Conference on Computing Education Research (Koli Calling '23), November 16–19, 2023, Koli, Finland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Universities are increasingly adopting automated feedback in Computing courses by using Automated Assessment Tools (AATs) rather than Teaching Assistants (TAs) in an attempt to scale educational resources and reduce the cost of providing students with individualized feedback [29, 38, 41]. However, little research quantifies the various benefits and consequences of using automatic feedback (using computerized AATs) over manual assessment (using human TAs). Investigating how automatic and manual feedback impact students in isolation and together is crucial to understand the implications of particular design decisions and economic choices. Thus, this study quantifies and compares the effects of providing *only* automatic feedback, *only* manual feedback, and *both* automatic and manual feedback for N=117 students enrolled in our bachelor programme *Software Development*. These three conditions naturally arise when adopting an institutional resource perspective, i.e., it boils down to the choice between a human teaching assistant versus computerized (virtual) teaching assistants. For providing students with feedback on their programming assignments, an educational institution can either employ potentially several TAs or have a single person set up and superintend an automated teaching assistant—or a combination of both manual and automated feedback.

We perform a controlled experiment that exposes all students to the same programming exercise, but partition them into three groups corresponding to the aforementioned three conditions. We compare the objective task effectiveness of students' solutions in terms of *correctness*, *duration*, and *code smells* as well as subjective student perspectives in terms of *frustration*, *assistance*, and *preferences* to provide educators with the data necessary to make actionable evidence-based recommendations regarding the inclusion of automatic feedback. We consider both objective task effectiveness and a subjective student perspective to account for quality (using multiple parameters) and student well-being.

2 BACKGROUND & RELATED WORK

Automated assessment has been around for more than 60 years to assess students' assignments [7]. Researchers have compared 30 different automated assessment tools (AATs) and categorised them [30]. Common for all is that they provide instant feedback on student submissions, also being the primary argument for utilising such platforms [3, 11, 39]. Most of these tools are web-based (online) and support multiple languages [3, 30].

AATs usually focus on *functional* (aka, *input/output*) correctness and provide feedback from *summative* assessment (i.e., *of* learning); whereas teaching assistants (TAs) usually focus on overall *structural* correctness and provide feedback from *formative* assessment (i.e., *for* learning) [10].

Automated assessment can differ according to multiple parameters, Nutbrown and Higgins differentiate between *Combo tools* (combination of multiple automatic tools), *Style and structure tools* (checking beyond input / output), *Informative and specific* (e.g., producing student advice), *Timely* (The time taking for the tool to provide feedback), *Reliable and consistent* (Both summative and formative feedback should be reliable and testable), *Clearly communicated* (Feedback should be understandable), and *useful for teachers* (Feedback and assessment is not only for the students) [28]

Separately, Ihantola et. al. categorise AATs into either tools for programming competitions or (introductory) programming education. [22] These various attributes can be realised in different ways, e.g., building a custom email client on top of existing platforms or using tools with the built in capability of checking for patterns using regular expressions, such as *CourseMarker* [28].

On the use of Automated Assessment. Research reveals that AATs positively impact student learning [3, 32]. However, this is primarily measured by end-of-course grades and is not further explored on other parameters [3]. Also, research finds that automated assessment platforms, such as the platform Kattis, release the burden of marking/assessing programs and make students feel more confident about their solutions [12]. Additionally, automated feedback has the benefit of low turnaround time and fairness which are all concerns of manual feedback. Thus, with cohort sizes increasing, reducing time is a substantial motivation [28].

A different, less obvious benefit, is that programming exercises are a key source in the high levels of student stress, dissatisfaction, academic dishonesty, low grades, and high drop out rates [5]. Automatic assessment tools are one way of alleviating some of these issues through easy creation and grading, quick and accurate feedback to students, freeing of instructors' time, and enabling many small programming problems which are more digestible for the students [5]. And these many small problems or coding exercises do not impact students' stress levels as they take less time to complete from a student perspective [16].

Furthermore, when using automatic feedback some platforms support activities seldom seen within manual assessment such as impacting marks based on violations of particular standards or requirements. Something which is difficult with manual assessment due to human errors [28].

On the use of TAs. Separately, research finds that properly trained TAs may positively impact students' academic performance [13]. Consequently, research outlines ideas for training, best practices, and various roles of TAs [21, 35]. Yet, according to a literature review by Mirza et al. these best practices have yet to be rigorously researched [27]. As the training of TAs varies greatly, it is intrinsically difficult to generalize their impact on student problem-solving [34]. While mostly relying on anecdotal evidence, TAs are effective at providing detailed feedback tailored to the individual, and increase student satisfaction, moral, attitude, and motivation [27]. Furthermore, there is no accepted definition of an expert human TA. Yet, TA's experience in teaching, as long as they are subject matter experts and have the same base training, does not appear to impact effectiveness [40].

Automated vs Manual Assessment. Automated assessment tools provide the benefit of low turnaround time and fairness which are both concerns of manual feedback. Thus, with cohort sizes increasing, reducing time is a substantial motivation [28]. Prior research argues for combining both manual and automated as TAs can provide additional feedback (or override grades), e.g., what *Web-Cat* does [22].

Despite the obvious benefits of automated assessment, prior research has found contradicting results on its use. Considerable divergences were found when grading using automated assessment and a manual baseline on 77 assignments; automated assessment awarded more top marks, but also a lot more failures (in fact, almost half of the students (37:77) received a score of $\leq$30 from automated assessment whereas this was only the case for 8% (6:77) using manual assessment) [33]. Ultimately, automated assessment was deemed *unreliable* for grading purposes. In contrast, Gordillo finds that automated assessment improves student motivation, quality of their work, and enhances practical programming skills, although students find the feedback hard to understand [18]. Additionally, Alemán found a positive effect in regards to debugging, deployment, and versioning but not testing when students are using an AAT [4].

Research Gap. The prior work was in the context of grading of submitted solutions and provide contradicting results, whereas our work focuses on feedback during exercise classes. Our work compares the use of a particular automated assessment tool with the use of teaching assistants to better understand the interplay between automated and manual assessment for *formative* assessment. While a variety of AATs exist (e.g., CodeLab [26], CodeJudge [24], JavaAssess [23]), we have chosen the tool Kattis [2] as it is widely adopted at our University. Kattis allows users to upload solutions to individual programming problems in multiple programming languages using an online service. The platform then evaluates the code according to a predefined set of unit test cases for the given problem and provides the student with binary feedback; i.e., *pass* or *fail.* We have deliberately chosen Kattis and to provide the students with limited quantitative pass/fail-feedback to study the *extremes* along the spectrum from rich qualitative feedback (provided by a human) to limited quantitative feedback (provided by a tool). For other tools, providing more human-like feedback, we expect them to behave interpolationally between the extreme end-points of our study.

3 METHODOLOGY

We detail the setup of our controlled experiment.

3.1 Objectives

Our experiment is organized around two complementary research questions. The first is focused on (objective) *task effectiveness*; the second on (subjective) *student perspectives*:

RQ1 (Task Effectiveness): How do *TAs vs AATs compare* for providing feedback on student programming assignments in terms of **task effectiveness** such as *correctness*, *duration*, and *code smells*?

RQ2 (Student Perspectives): How do *TAs vs AATs compare* for providing feedback on student programming assignments in terms of **student perspectives** such as *frustration*, *assistance*, and *preferences*?

In concert, the two research questions provide a complementary perspective on the advantages and disadvantages of using teaching assistants vs automated assessment tools, such as Kattis, for providing feedback on student programming assignments.

3.2 Participants

The experiment takes place in the context of the three-year Bachelor of Software Development (BSWU) at the IT University of Copenhagen (ITU). The programme does not require prior programming skills and admits approximately 150 students each year. In 2022, there were 153 students (hereof 20% women: 30 out of 153). 33% had little or no prior programming experience; 45% had some limited experience; and 22% had prior programming experience. The first semester of the programme consists of three simultaneous courses: Introductory Programming (CS1), Discrete Mathematics, and Project Work & Communication.

Our experiment was conducted a month into the 15 ECTS[1] first-semester CS1 course which consists of seven weekly mandatory Kattis programming assignments, three mandatory one hour closed-book on-premises programming tests, and two mandatory hand-ins that must be approved before they are eligible for the exam. The course utilises the BlueJ IDE. The automated assessment platform Kattis is used for seven mandatory assignments. Our experiment uses the fourth mandatory Kattis programming assignments as the task the students need to solve. The students will therefore have used the platform minimally before the execution of the experiment. However, this way the students get something out of participating in the experiment. Similarly, all students have prior experience with utilising teaching assistants. The experiment is conducted as part of their regular teaching activities. On the day of the experiment, **N=117** students participated (hereof 25 women and 44 students with little to no prior programming experience).

3.3 Task

For experimental task, we settled on the `FizzBuzz` Kattis problem (translated into a Danish version, '`ØfGrynt`') since it exercises student abilities in implementing nested `if-else` statements using non-trivial boolean conditions. Besides, this problem had been in use in previous editions of the CS1 course, constituting the fourth mandatory Kattis programming problem.

A solution to the `FizzBuzz` programming problem is supposed to output the first 1,000 integer numbers consecutively starting from *one*, but where each number divisible by 3 is replaced by the word '`Fizz`' and each number divisible by 5 by '`Buzz`'; numbers divisible by both three *and* five (by 3×5) are replaced by the concatenation: '`FizzBuzz`' (hence the name of the problem). Additionally, counting has to *restart* from one every time the count hits 100. Figure 1 horizontally lists an initial part of the intended output sequence.

The problem had earlier been made into a Kattis exercise specifically for students taking another CS1 course by an independent teacher[2] and is inspired by the '`FizzBuzz`' problem used in many programming interviews [17].

For assessment of task fulfillment, the Kattis problem comes with two sets of unit test cases (withheld from the students): *partial* correctness, comprises nine test cases; and *full* correctness adds an additional five test cases with more corner cases. For the CS1 course, *partial* correctness is enough to get the mandatory problem approved, but most students strive for *full* correctness.

3.4 Treatments

Our controlled experiment considers two treatments:

TAs: The first treatment is to use *teaching assistants* for providing feedback on student programming assignments. During exercise classes, the role of a teaching assistant is usually to offer *formative* feedback along the lines of a *coach*. Typically our TAs aid students when errors arise or comment on the quality of the code written. The students were free to ask as many questions to the TAs as they wanted, only inhibited by the natural scarcity of shared limited resources. All five TAs (three women, two men) were highly qualified; in fact, they had been selected from a pool of 27 qualified applicants. The TAs were given a correct solution a few days in advance, in order for them to have adequate time to prepare to assist the students with the programming problem.

AAT: The second treatment is to use an *automated assessment tool*, specifically Kattis, for providing feedback on student programming assignments. The role of the automated tool is usually to offer *summative* feedback to the student (and teachers) along the lines of a *judge*. The students were free to upload their solution as many times as they wanted to the Kattis platform which then responded with one of six verdicts: 1) *compile-error*, 2) *runtime error*, 3) *time-limit exceeded*, 4) *incorrect*, 5) *partially correct*, or 6) *fully correct*. Hence, the tool provided limited quantitative feedback.

3.5 Conditions

For comparing the two treatments TAs vs AAT for assessing student programming assignments, we use three conditions:

- **TAs:** Students will have *only* TAs at their disposal for feedback on their programming assignment;
- **AAT:** Students will have *only* an AAT at their disposal for feedback on their programming assignment; *and*
- **TAs+AAT:** Students will have *both* TAs *as well as* an AAT at their disposal for feedback on programming assignment.

The advantage of using these three conditions (where TAs as well as AAT are subsumed by TAs+AAT) is that it makes it possible to quantify the effect of *adding* TAs as well as that of *adding* an AAT as a form of student feedback. The conditions { TAs, TAs+AAT } attest to the effect of having TAs (irrespective of an AAT); whereas the conditions { AAT, TAs+AAT } account for the effect of having a AAT (irrespective of TAs).

[1]One academic year is 60 ECTS (European Credit Transfer and Accumulation System).

[2]Developed by Prof. Thore Husfeldt; problem available from https://itu.kattis.com/.

```
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, 16, 17, Fizz, 19, ...
```

Figure 1: The "FizzBuzz" problem where every 3rd number is replaced by the word 'Fizz' and every 5th by 'Buzz'; although, the output is shown here with a comma rather than a newline. (Translated from Danish where 'Fizz' = 'Øf' and 'Buzz' = 'Grynt'.)

3.6 Design

All 135 students were randomly assigned to one of the three conditions. However, due to only N=117 students being present on the day of the experiment, the three conditions, TAs, AAT, and TAs+AAT; each received N_1=39, N_2=42, respectively, N_3=36 students. Each condition was assigned a dedicated room. The five regular TAs from the CS1 course were assigned as TAs; two TAs (one woman, one man) for the TAs condition; two TAs (one woman, one man) for the TAs+AAT condition, and a single TA (woman) for the AAT condition to simply passively monitor that the students adhered to the rules of the experiment. The amount of TAs accessible to students within the TAs and TAs+AAT conditions simulate the number of TAs accessible during a regular lab session. (In our Danish context, usually around a 1:20 assistant-*to*-student ratio.)

The programming assignment was individual, so the students were not allowed to interact with each other during the experiment. The students, however, did have access to their notes, teaching material, and access to the internet. (They were asked to not communicate with each other online. Note also that the experiment took place before the proliferation of ChatGPT.)

The experiment began with a TA reading instructions that explained the rules of engagement. Hereafter, the students had one hour to finish the task. For each student, we recorded a number of metrics (see below). Upon finishing the task, the students had to complete a survey (see further below).

3.7 Pilot Study

Before runnning the experiment, a pilot trial was performed with six students who were taking a similar CS1 course, but on another educational programme (Software Design). Two of these students had some prior programming experience. Both CS1 courses used Java and their learning goals were almost identical. The pilot concluded that the KATTIS problem was likely fitting for the intended target audience of novice programmers with no or limited prior programming experience. However, it became clear that the problem specification needed to be further elaborated.

3.8 Metrics

For each student's final submission, we recorded the *correctness* as the percentage of successful unit tests (and whether or not the code, in fact, compiled or ran). We recorded the total *duration* of the time spent working on the programming assignment. Finally, we ran SonarCube [37] (using default settings) on the student solution to obtain the number of warnings (aka, *code smells*) based on analyzing, e.g., naming conventions, indentation, and unused code. None of this information was passed on to the students, but used solely for this study as a superficial proxy assessing the code quality of the solutions handed in.

3.9 Survey

Upon finishing the experiment, students received a questionnaire which asked the students about their level of *frustration* and the level of *assistance* available to them during the programming exercise session, both on a 1–5 Likert scale. Students were also asked about their *preferences*; whether they preferred feedback from a TA or an AAT, or if they did not have any preference. Finally, students were also invited to provide qualitative feedback elaborating on their ratings.

3.10 Ethics

We solicited and obtained ethical approval from our University to carry out this experiment as well as explicit informed consent from the students to participate in the study. All students had a chance to opt-out of the experiment (whereby we would not record any data for them); however, all N=117 students present on the day of the experiment consented. All data has been anonymized and is GDPR complaint. We are not interested in the performance of individual students; only in the "big picture," comparing the three conditions of the experiment (based on anonymized data from around 40 students in each condition). Differential treatment of the students was limited to this one hour experiment which differed only in the assistance available to the students while working on the `FizzBuss` exercise. To minimize the differential treatment, an additional lab session was offered to those in need (with both KATTIS and TAs available) with only a few attendees (from all treatments). We did not include an unassisted feedback condition as this would (in most cases) force the students under this condition to attend the additional lab session. Student grades in the course did not, in any way, depend on their performance in this experiment.

3.11 Analysis

For comparing *vector data* between two populations (e.g., correctness percentages or Likert scale preferences for TAs vs TAs+AAT), we used the (Mann-Whitney) U-Test[3] which does not presuppose that the data is normally distributed.

For comparing *proportional data* (e.g., ratios of students prefering feedback from TAs vs AAT), we used the Z-Test[4] (for two population proportions) which is capable of comparing whether one ratio, x_1/y_1, is statistically significantly greater than another, x_2/y_2 (whenever: $x_i \geq 5$ and $y_i - x_i \geq 5$, for $i \in \{1, 2\}$). Whenever the previous condition is not fulfilled, we use Fischer's Exact test[5] which is capable of dealing with smaller sample sizes.

For all tests, we adopt a conventional 95% confidence interval (α = 5%) and perform only symmetrical two-tailed tests which do not presuppose the superiority of either testee in question. Occasionally,

[3]U-Test: https://www.socscistatistics.com/tests/mannwhitney/
[4]Z-Test: https://www.socscistatistics.com/tests/ztest/
[5]Fischer's Exact Test: https://www.socscistatistics.com/tests/fisher/default2.aspx

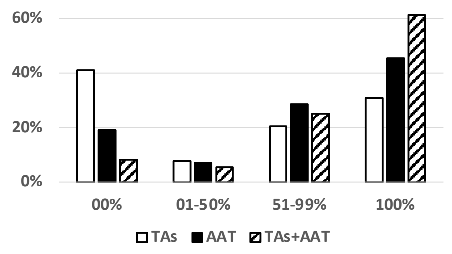

Figure 2: Correctness: Percentage of successful unit test cases for students under the three conditions. The difference between TAs vs TAs+ATT is significant (p=0.0024).

we will refer to *weak* significance using a 90% confidence interval (α = 10%) which is not unreasonable outside of the biomedical domain.

4 RESULTS

We consider the results according to our two research questions:

4.1 Objective Task Effectiveness (RQ1)

Correctness is defined as the *percentage* of unit test cases that a student's solution passes successfully, using the *full* correctness suit of unit test cases; i.e., all 14 (9+5) test cases (cf. Section 3.3).

> **Observation 1a (Correctness):** Students with access to feedback from *both* AAT *and* TAs make significantly *more correct* solutions than those with access to feedback from only TAs.

Figure 2 shows the distribution of percentages of successful unit test cases for students according to the three conditions. To the far right, we see that more than 60% of the students with access to feedback from both AAT *and* TAs write code that passes all unit test cases. This ratio drops to 45% for students with access to feedback from only AAT and, further, to around 30% for students with TA-feedback only. To the far left, we see an inverted picture for students *not* getting a single unit test case right. In fact, *half* of the 16 students not getting any unit test cases right under the TAs condition, hand in code that does not even compile. Students with access to AAT (irrespective of whether there were TAs present) only handed in compiling code (with one exception under the AAT condition). After all, making sure the code compiles (and runs) is precisely one among many intended benefits of an AAT.

The difference between the *correctness* scores for students under the TAs vs TAs+AAT is statistically significant (p=0.0024); whereas the other pairwise differences are not ($p \geq 0.072$).

Duration is quantified as the time spent working on the Kattis programming assignment. Students received an hour for the task; in the end, the students received an extra 10 minutes (since many were close to finishing the task).

> **Observation 1b (Duration):** Students handing in the final submission with access to feedback from both AAT *and* TAs are significantly *faster* than those with access to feedback from only TAs.

Figure 3 shows a boxplot of the distribution of time spent working on the assignment under the three conditions. The *top* and *bottom* horizontal lines represent the *longest* and *shortest* time expenditure; the *top* and *bottom of the square boxes* represent the *upper* and *lower* quartiles; the horizontal *line inside the box* represents the *median*; finally, the *plus* designates the *mean* (*average*). From left to right, we see a consistent drop in the *median* time from 51'29" to 46'30" to 34'21". (The *mean* time spent exhibits a similar drop from 47'16" to 42'25" to 38'38".)

Again, statistical analysis reveals that the pairwise difference between TAs and TAs+AAT is significant (p=0.028); whereas the other pairwise differences are not ($p \geq 0.28$).

Code smells are measured as the number of code smell warnings reported by SonarCube. We use code smells as a superficial proxy for the *code quality* of student solutions.

> **Observation 1c (Code Smells):** Students with access to feedback from both AAT *and* TAs write code with significantly *fewer code smells* than those with access to feedback from only TAs.

Figure 4 shows the distribution of the number of code smell warnings under the three conditions. To the left, we see that more than 40% of the students under the TAs+AAT condition have four or fewer code smells; under AAT, this number drops to around 30% and to 20% under the TAs condition. We see the opposite pattern when it comes to code with lots of warnings (see the right hand side of Figure 4).

Once again, statistical analysis reports that the pairwise difference between TAs and TAs+AAT is significant (p=0.015); whereas the other pairwise differences are not ($p \geq 0.13$).

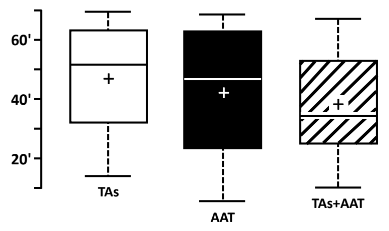

Figure 3: Duration: Distribution of time spent working on the AAT programming assignment under the three conditions. The difference between TAs vs TAs+ATT is significant (p=0.028).

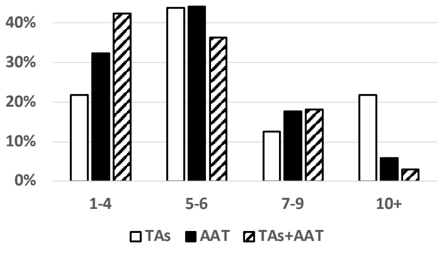

Figure 4: Code smells: Number of warnings from SonarCube under the three conditions. The difference between TAs vs TAs+ATT is significant (p=0.015).

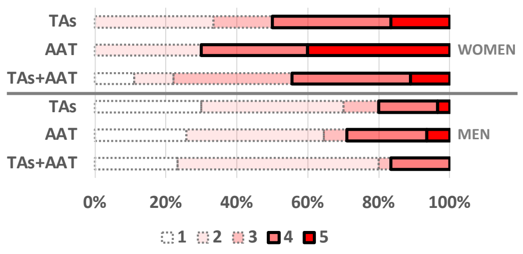

Figure 5: Frustration: Student frustration levels (1–5) under the three conditions. Darker shades of red (higher scores) indicate higher levels of frustration. In general, women are *more frustrated* than men (p=0.00008). There appear to be *more maximally frustrated students* under AAT than the other conditions (p=0.068).

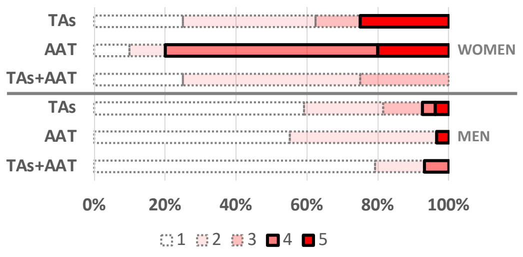

Figure 6: Assistance: Student unmet assistance needs (1–5) under the three conditions. Darker shades of red (higher scores) mean more unfulfilled assistance needs. Women are, in general, more affected than men (p<0.00001) as are women with feedback from only a computer (AAT) than those with feedback from (also) a human TA (p=0.0083).

Non-Dominant Perspective. In order to ensure that non-dominant sub-groups are represented and not "diluted" into the data, we considered *women* and programming *inexperienced* students (those starting CS1 without prior programming experience). For RQ1, the results were similar when projecting onto either the 25 women or 44 students with no or little prior programming experience. The statistics revealed only *weak* significance (e.g., p=0.059 for *correctness* & p=0.087 for *duration* when zooming in on the women); however, this is presumably due to the lower number of data points. Importantly, students without prior programming experience were offered an intensive 3-day CS0 programming onboarding course which, according to recent research, is sufficient to bridge the prior experience gap [20]. Also, the experiment took place one month into the CS1 course, further diminishing the prior experience gap.

4.2 Subjective Student Perspectives (RQ2)

For RQ2, there were considerable differences between the perspectives of women vs men; hence, we have split the N=117 data points into: 25 women & 92 men. (There were much less differences between students when considering prior programming experience.)

Frustration is quantified as a student self-reported number on a 1–5 scale where a rating of 1 is *absence* of frustration and a rating of 5 is the *highest* level of frustration with the exercise session.

> **Observation 2a (Frustration):** Women are, in general, significantly *more frustrated* than men when solving the Kattis problem, independent of the form of feedback. Also, there appear to be *more maximally frustrated students* when having access to only automated feedback than when (also) having access to manual feedback from a human TA.

Figure 5 shows the levels of frustration under the three conditions for women (top) vs men (bottom). We see that, in general, women are significantly more frustrated than men (p=0.00008). Also, both genders are most frustrated under the AAT condition; in fact, 70% of the women reported *high* levels of frustration (a frustration rating of 4 or 5) when having access to feedback only from a computer (ATT). The statistical analyses, however, report that the ratio of students exhibiting the highest level of frustration (rating 5) is only weakly significant (p=0.068) when comparing the AAT condition versus the two other conditions. TAs+AAT appears to be better than the two other conditions, although the differences are not significant ($p \geq 0.29$).

Assistance is measured as a student self-reported number on a 1–5 scale where a rating of 1 corresponds to students getting all the assistance needed; a rating of 5 means that they required a lot more assistance than what was available.

> **Observation 2b (Assistance):** Women, in general, report significantly more unmet assistance needs than men; in particular, women with access to only automated feedback assistance are significantly more affected than women who (also) have access to manual feedback assistance from a human TA.

Figure 6 shows the unmet assistance needs under the three conditions for women (top) vs men (bottom). We see that women, in general, have significantly more unmet assistance needs than men (p<0.00001). Also, significantly more women report *high* unmet assistance needs (rating 4 or 5) when having access to feedback only from a computer (ATT) compared to the two other conditions (p=0.0083). For men, there were no significant difference between those under AAT vs the other conditions (p=0.66). TAs+AAT appears to be better than the two other conditions, although the differences are not significant ($p \geq 0.15$).

Preferences. We asked the students to choose whether they preferred to receive feedback from: a (human) TA, a (computerized) AAT, or if they did not have any preference.

> **Observation 2c (Preferences):** Women, in general, prefer access to feedback from a human TA over a computer (AAT); in contrast, men, in general, do not exhibit a preference for receiving feedback from either a human (TA) or a computer (AAT).

Figure 7 shows the preference for receiving feedback from a human (TA) vs a computer (AAT). For women, in general, we see a strong preference for receiving feedback from a *human* over a *computer* (p=0.0023). For men, in general, there is no clear difference (p=0.44). Because of *diversity* (different students have different needs and want different things), the TAs+AAT condition is superior since it incorporates both forms of feedback: manual (human) *and* automated (computer). These differential gender preferences

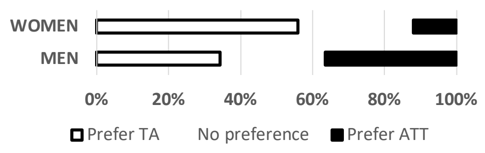

Figure 7: Preferences: Student preferences for getting feedback from TAs vs AAT. The overall difference is not significant (p=0.368).

Cost:	Measure	TAs	AAT	TAs+AAT
Resources	Time (effort)	$a \cdot \|S\|$	b	$a \cdot \|S\| + b$
	Asymptotic	$O(\ \|S\|\ )$	$O(\ 1\ )$	$O(\ \|S\|\ )$

Benefit:	Measure	TAs	AAT	TAs+AAT
RQ1	Correctness	−		+
	Duration	−		+
	Code Smells	−		+
RQ2	Frustration		−	+
	Assistance		−	+
	Preferences		−	+

Figure 8: Cost/benefit trade-off summary of the different forms of feedback. Cost is summarized at the top, benefit at the bottom; |S| is the number of students in the given course. '+/−' denotes statistically significantly best/worst; whereas '+/−' (gray font) means best/worst, but non-significantly so.

on *human* vs *machine* feedback appear to coincide with recent research demonstrating that women, in computing, in general, prefer tasks involving *people* over *things*, whereas men, in computing, in general, do not exhibit preferences along the *people–things* spectrum [8, 19, 25].

5 DISCUSSION

We first consider the cost/benefit trade-offs associated with each of our three experimental conditions, and then use the implication of our findings to issue recommendations for educators on how to choose between different forms of providing students with feedback on programming assignments.

5.1 Cost/Benefit

Figure 8 provides an overview of the *cost* and *benefit* under each of the three conditions (cf. the three rightmost columns): (only) TAs, (only) AAT, and (both) TAs+AAT.

Cost. The top part of the figure quantifies cost in terms of educational institutional resources. In the top row, cost is specified in terms of temporal resources 'Time (effort)' using coefficient variables: a for per-student temporal cost and b for constant temporal cost (as in, e.g., $a \cdot |S| + b$, where $|S|$ denotes the number of students). The bottom resource cost row gives the *asymptotic* cost complexity (as a function of the number of students, $|S|$). Asymptotic complexity is particularly important for reasoning about the *scalability* of the various assignment feedback solutions available.

In the following, tasks common to all conditions, such as *selecting an appropriate assignment* and *formulating a problem specification*, have been abstracted away, since they would simply add additional constant factors to the resource cost under all conditions.

Under the TAs condition, all TAs are collectively tasked with: T1) *read/solve the assignment* (with a constant cost for each of the TAs which is proportional to the number of students); *plus* T2) *provide formative feedback* for all student solutions (with a per-student cost; i.e., $O(\ |S|\ )$). In total, this amounts to a linear cost: $a \cdot |S|$ which is in the $O(\ |S|\ )$ complexity class.

Under the AAT (AAT) condition, the tasks are: T1') an educator (teacher or TA) has to *set up the AAT* once and for all (with a constant cost); *plus* T2') the AAT has to *provide summative feedback* of all student solutions (which is constant because of automation). Overall, this amount to a constant cost: b in $O(\ 1\ )$. In practice, there may be a negligible per-student overhead (monitoring student progress); ideally, however, even this may be automated.

When both TAs and an AAT are accessible (the TAs+AAT condition), the cost will be the *sum* of the costs of both conditions with a cumulative linear cost: $a \cdot |S| + b$ in $O(\ |S|\ )$.

Benefit. The middle and bottom part of Figure 8, summarizes the *benefits* according to the results (Section 4), grouped by the two research questions: RQ1 (objective task efficiency) and RQ2 (subjective student preferences). The black symbols (+/−) denote statistically significant evidence of '+' being statistically significantly superior to '−'. The gray symbols (+/−) means better/worse, but not statistically significantly so. (For RQ2, gender is taken into account when summarizing the better/worse (+/−) verdicts.)

For RQ1, all task efficiency metrics painted a consistent picture: access to feedback from both TAs and an AAT is statistically significantly superior to feedback from TAs in terms of *correctness*, *duration*, and *code smells*. Also, all metrics consistently placed access to feedback from both TAs and an AAT almost exactly in between the two other conditions. Thus, it appears that TAs and an AAT is better than AAT (although none of the pairwise tests between these *smaller differences*, involving the AAT condition, exhibited statistical significance).

For RQ2, *frustration* levels were weakly significantly worst under the AAT condition; also, the TAs+AAT condition appeared to be best, but not significantly so. In term of (lack of) *assistance*, women were worst off under the AAT condition; also, the TAs+AAT condition appeared to be best, but not significantly so. Finally, in terms of *preferences*, women significantly preferred a TA over an AAT which means that AAT is worst and the combination of both (TAs+AAT) is trivially best since it incorporated both forms of feedback: students are free to choose for themselves wherefrom they want feedback.

5.2 Four Recommendations

Given the findings generally demonstrating complementary advantages of automated vs manual feedback, we can distill optimal scenario equilibria which appropriately balance trade-offs between costs and benefits, depending on the number of students and the per-student resources available at the given educational institution.

Both TAs + AAT is optimal whenever there are enough students for the overhead of introducing an AAT to pay off (from automation) as well as an abundance of (per student) resources:

Recommendation 1 [↑Resources]**:** Educators with *many resources* **should consider** providing students with access to *both* instantaneous feedback from automated assessment *as well as formative* feedback from manual assessment (human TAs).

Synergetically, having access to feedback from *both* manual *as well as* an AAT is superior both from a task effectiveness and a student perspective; in concert, the two forms of feedback essentially provide "the best of both worlds." After all, the AAT plays the part of a *judge*, while a TA plays the role of a *coach*. Access to both comes with complementary advantages. This is also advantageous because of diversity since different students have different preferences. A case study of using automated assessment (during the Covid-19 pandemic) found that if automated assessment is used as an optional add-on, the majority of students would use it [6].

However, this solution compromizes on cost; not all educational institution will be able to "afford the luxury" of providing access to feedback from both TAs as well as an AAT. We now consider three enumerable options that cut on cost, but, in turn, each compromize along different dimensions.

Only AAT appears to be optimal whenever there are many students and fewer institutional resources; then, many students can be provided with feedback at a low per-student cost; however:

> **Recommendation 2** [↓Resources & ↑Students]**:** Educators with *few resources* and *many students* **should be wary of** providing students with access to only instantaneous feedback from automated assessment; although it provides scalability at low cost, it will affect women negatively in terms of *frustration*, *assistance*, and *preference*.

This solution is often adopted for Massive Open Online Courses (MOOCs) where low-cost scalablity is critical [38]. However, it compromizes on student perspectives (RQ2). While the negative effects may be tolerable by men (see bottom parts of Figure 5, 6, & 7), women are significantly negatively affected in terms of *frustration*, *assistance*, and *preference* by not having access to human feedback (see top parts of Figure 5, 6, & 7).

Only TA is optimal whenever there are few students and institutional resources; then, there is no overhead of introducing an AAT and few students can easily be accomodated by a (part-time) TA:

> **Recommendation 3** [↓Resources & ↓Students]: Educators with *few resources* and *few students* **should consider** providing students with access to only *formative* feedback from a (part-time) human TA, proving personalized feedback without any overhead from automation.

Small classes may benefit from individualized assistance [9]. However, this solution compromizes on task effectiveness (RQ1); in particular, on student solution *correctness*, the time students spend on assignments (*duration*), and superficial indicators of code quality (*code smells*). (In countries with access to cheap student labor, it may also be possible to use TAs for scalability [15], but this depends on the cost of TAs which varies a lot between countries [1].)

Scalable Compromize. A pragmatic combination exists: It is possible to harness automation for scalability, while providing human feedback from a very limited (sub-linear) pool of TAs assigned to assist only the students mostly in need of individualized human feedback and guidance:

> **Recommendation 4** [→Resources & ↑Students]**:** Educators with *moderate resources* and *many students* **should consider** providing students with access to instantaneous feedback from automated assessment as well as access to *formative* feedback from a very limited pool of TAs assisting only the students mostly in need of human feedback and guidance.

For true *low*-cost *high*-student-volume scalability, to several hundreds or even thousands of students (including MOOCs [31]), the number of TAs, $|T|$, may have to be *sub-linear*; e.g., $|T| = \log(\,|S|\,)$. Pragmatically, having a limited constant pool of TAs ($|T| = k$ or 1), may go a long way to accommodate the students mostly in need of a human feedback or guidance.

Please note that the choice of which of the four recommended scenarios to opt for is highly context-dependent; it also depends on teaching philosophies; in particular, trade-off of whether to optimize for task effectiveness (RQ1) vs student perspectives (RQ2) and the willingness of an institution to accommodate for diversity.

Overall, our findings and recommendations contribute with data and evidence, including estimations of the size of the effects involved. We direct our research to educators and educational institutions; the *actionable evidence-based recommendations* ought to be straightforward for educators and institutions to heed and operationalize.

6 THREATS TO VALIDITY

First, we consider methodological soundness of making conclusions based on our study involving a single assignment (*conclusion validity*). Second, we scrutinize how the metrics were obtained and measured (*construct validity*). Third, we investigate biases towards the results obtained (*internal validity*). Fourth and finally, we ponder the extent to which our findings generalize (*external validity*).

6.1 Conclusion Validity

Study based on *one* assignment? Our study involved only one assignment; namely, FizzBuzz. However, it is a fairly prototypical exercise involving integer arithmetic, Boolean conditions, nested if-statements, output, and bounded iteration. It appears to be a very common programming exercise problem; in fact, a Google search of `"FizzBuzz programming exercise"` yields more than 40K search results (as of March 2023). Futher, suffixing the search query with `" in "` makes Google's auto-completion suggests numerous programming languages (including C, Python, Java, and JavaScript). We therefore expect that our findings would be relevant for other, similarly difficult introductory programming assignments. (For more complex exercises, we refer to Section 6.4.) Of course, there is an inherent risk in extrapolating from a single assignment, but that does not invalidate conclusions from single-case study research [14].

6.2 Construct Validity

Measuring duration? Time spent on the Kattis assignment is measured as the difference between the *start time* (same for all students) and the *end time* (when a student began filling out the online survey). Students were instructed to start the survey as soon as they finished the task; this was also written on the big screen in each room and monitored by the TAs. Students were only allowed to leave the room upon finishing the survey questionnaire.

Measuring gender? Participant gender information was obtained from the enrollment system based on information from the

Danish central person registry (CPR). The gender in the registry is binary, but people can have their gender information changed to reflect their self-identification.

6.3 Internal Validity

Correctness biased towards the AAT? The AAT served two purposes during the experiment: First, it was the *feedback* treatment that two out of three of the students were exposed to; second, it was also the tools used to *assess* the correctness of solutions. While this is the natural usage of an automated assessment tool, it does, by design, favor the conditions involving the AAT.

Students understood role of TAs? Students exposed to the TA treatment were informed that they were allowed to ask for as much help as needed. Due to social anxiety, some students find it difficult to ask others for help. To accommodate this, the TAs reminded students that they were allowed to ask questions as in a typically exercise class (outside the experiment). We argue that the situation is realistic in that it mimicks regular exercise classes. This is also precisely why some students might prefer to receive feedback from an automated tool over a TA.

Student interaction? During the experiment, students were not allowed to interact and were informed of this before the experiment started. This is because we wanted to record data for each *individual* student, independently. Technically, malicious students could, in principle, have communicated online, but since the stakes were low and they were informed that this constituted research, we do not believe this played any role in the experiment. The TAs also monitored that students followed the rules.

TA comparability? Differences among the TAs may introduce comparability biases. Note, however, that all TAs passed the TA hiring criteria for CS1. All were competent, had TA experience before the experiment, and were selected from a large pool of 27 qualified applicants. Also, we made sure that the two conditions were feedback could be provided by a TA (i.e., { TAs, TAs+AAT }) both has a woman and a man assistant TA to mitigate any TA gender effects.

Different rooms? Due to limited room availability, two of the conditions took place in (inclined) auditoria, while the third (TAs) took place in a regular (flat) classroom. We expect this to have minimal impact on the experiment especially since the students have previously been working on assignments in both types of rooms.

6.4 External Validity

Beyond Kattis? Our study used Kattis as the AAT, deliberately configured to provide only a binary *pass/fail* level of information, without revealing anything about which unit test cases may have failed. Other automated assessment tools provides this information to the student. We settled on Kattis as the tool is widely adopted in our organisation and to compare the two extremes: TAs vs minimal *pass/fail* feedback. It could be interesting to replicate the study with an automated assessment tool that provides more feedback to the student and investigate how this impacts the findings. We expect frustration levels may drop if a tool provides more, especially human-like, feedback.

Beyond simple programming tasks? The experiment tested the student with one specific Kattis problem that most students managed to solve within the designated one hour time limit. The '`FizzBuzz`' problem has a difficulty level of 1.7 (for context, '`Hello World`' is 1.2, and the most challenging level is 9.6). We expect our results to generalize to similarly simple tasks intended for programming novices. We expect that harder tasks would make access to feedback from TAs even more of a limited shared resource to the point where it may start becoming a "bottle neck." Similarly, we expect frustration related to the AAT increases as the students spend more time making their solution pass, leading to more failure responses from the platform. However, this is speculation; it would be interesting to see how the findings fare in the face larger more difficult tasks.

Beyond lab setting? The study was conducted as a *controlled* experiment in a "lab setting" which may have affected student behavior. Our experiment essentially optimized *internal* validity at the expense of *external* validity [36]. It would be interesting to follow students over a longer period, in a more realistic setting.

Beyond Software Development? Since there is nothing inherent to Software Development in our experiment, we expect the results generalize to any education for which programming plays a major role; including, Computer Science and Software Engineering.

7 CONCLUSION

We conducted a controlled experiment to quantify and compare the differences between two extremes: rich qualitative feedback (from a human TA) versus limited quantitative instantaneous feedback (from a tool) in the context of programming assignments. Our results demonstrated that the combination of *both* manual feedback (using TAs) *as well as* automated feedback (using Kattis) was superior both in terms of objective task effectiveness (*correctness*, *duration*, and *code smells*) and subjective student preferences (*frustration*, *assistance*, and *preferences*). Also, automatic feedback appears to be better in terms of task effectiveness; whereas manual feedback (using only TAs) appears to be better in terms of student perspectives. We uncovered significant gender differences related to subjective student preferences (*frustration*, *assistance*, and *preferences*) where women are negatively affected by not having access to feedback and guidance from a human TA.

The cost of providing manual feedback (in terms of TA resources) is linear in the number of students; whereas the cost of automated feedback is, in principle, constant (independent of the number of students). A cost/benefit analysis led to the identification of four equilibria which attempts to appropriately balance trade-offs between costs vs benefits of using manual versus automated feedback. Educational institution with many resources should consider providing manual as well as automated feedback. Institutions with few (per-student) resources should be wary of providing the students with access to feedback from only automated assessment as it significantly negatively affects women in terms of subjective student perspectives (*frustration*, *assistance*, and *preferences*). Institutions with few students should consider providing only manual feedback (avoiding the overhead of feedback automation). Finally, institutions with many students and moderate (per-student) resources, may consider the “scalable compromize” of using automated feedback for everyone and then a very limited pool of TAs assisting only the students mostly in need of human feedback and guidance. This strategy offers low-cost scalability while retaining the benefits of both automated and manual feedback.

These observations provide educators with evidence-based justification for budget requests.

REFERENCES

[1] [n. d.]. University Teaching Assistant Salaries by Country. https://www.salaryexpert.com/salary/browse/countries/university-teaching-assistant. Accessed: 2023-03-17.

[2] Aditi Agrawal and Benjamin Reed. 2022. A survey on grading format of automated grading tools for programming assignments. *arXiv preprint arXiv:2212.01714* (2022).

[3] Kirsti M Ala-Mutka. 2005. A survey of automated assessment approaches for programming assignments. *Computer science education* 15, 2 (2005), 83–102.

[4] José Luis Fernández Alemán. 2010. Automated assessment in a programming tools course. *IEEE Transactions on Education* 54, 4 (2010), 576–581.

[5] Joe Michael Allen, Frank Vahid, Kelly Downey, and Alex Daniel Edgcomb. 2018. Weekly programs in a CS1 class: Experiences with auto-graded many-small programs (MSP). In *2018 ASEE Annual Conference & Exposition*.

[6] Enrique Barra, Sonsoles López-Pernas, Álvaro Alonso, Juan Fernando Sánchez-Rada, Aldo Gordillo, and Juan Quemada. 2020. Automated assessment in programming courses: A case study during the COVID-19 era. *Sustainability* 12, 18 (2020), 7451.

[7] Julio C Caiza and José María del Álamo Ramiro. 2013. Programming assignments automatic grading: review of tools and implementations. (2013).

[8] Ingrid Maria Christensen, Melissa Høegh Marcher, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving People Rather Than Things Appeal More to Women (Recruitment Perspective). In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) *(ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 127–144. https://doi.org/10.1145/3446871.3469758

[9] Paul E. Dickson, Toby Dragon, and Adam Lee. 2017. Using Undergraduate Teaching Assistants in Small Classes. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 165–170.

[10] Dante D Dixson and Frank C Worrell. 2016. Formative and summative assessment in the classroom. *Theory into practice* 55, 2 (2016), 153–159.

[11] Stephen H Edwards. 2003. Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In *Proceedings of the international conference on education and information systems: technologies and applications EISTA*, Vol. 3. Citeseer.

[12] Emma Enström, Gunnar Kreitz, Fredrik Niemelä, Pehr Söderman, and Viggo Kann. 2011. Five years with kattis—using an automated assessment system in teaching. In *2011 Frontiers in education conference (FIE)*. IEEE, T3J–1.

[13] Peter Farrell, Alison Alborz, Andy Howes, and Diana Pearson. 2010. The impact of teaching assistants on improving pupils' academic achievement in mainstream schools: A review of the literature. *Educational review* 62, 4 (2010), 435–448.

[14] Bent Flyvbjerg. 2006. Five misunderstandings about case-study research. *Qualitative inquiry* 12, 2 (2006), 219–245.

[15] Jeffrey Forbes, David J. Malan, Heather Pon-Barry, Stuart Reges, and Mehran Sahami. 2017. Scaling Introductory Courses Using Undergraduate Teaching Assistants. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 657–658.

[16] Adam M Gaweda and Collin F Lynch. 2021. Student Practice Sessions Modeled as ICAP Activity Silos. *International Educational Data Mining Society* (2021).

[17] Imran Ghory. 2007. Using FizzBuzz to Find Developers who Grok Coding. https://imranontech.com/2007/01/24/using-fizzbuzz-to-find-developers-who-grok-coding/. Accessed: 2023-01-13.

[18] Aldo Gordillo. 2019. Effect of an instructor-centered tool for automatic assessment of programming assignments on students' perceptions and performance. *Sustainability* 11, 20 (2019), 5568.

[19] Pawel Grabarczyk, Alma Freiesleben, Amanda Bastrup, and Claus Brabrand. 2022. Computing Educational Programmes with More Women Are More about People & Less about Things. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 172–178. https://doi.org/10.1145/3502718.3524784

[20] Pawel Grabarczyk, Sebastian Mateos Nicolajsen, and Claus Brabrand. 2022. On the Effect of Onboarding Computing Students without Programming-Confidence or -Experience. In *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '22)*. Association for Computing Machinery, New York, NY, USA, Article 18, 8 pages. https://doi.org/10.1145/3564721.3564724

[21] Qiang Hao, David H Smith IV, Lu Ding, Amy Ko, Camille Ottaway, Jack Wilson, Kai H Arakawa, Alistair Turcan, Timothy Poehlman, and Tyler Greer. 2022. Towards understanding the effective design of automated formative feedback for programming assignments. *Computer Science Education* 32, 1 (2022), 105–127.

[22] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. 2010. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli calling international conference on computing education research*. 86–93.

[23] David Insa and Josep Silva. 2018. Automatic assessment of Java code. *Computer Languages, Systems & Structures* 53 (2018), 59–72. https://doi.org/10.1016/j.cl.2018.01.004

[24] Code judge. 2023. Code judge. https://codejudge.io. Accessed: 2023-01-13.

[25] Melissa Høegh Marcher, Ingrid Maria Christensen, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving People Rather Than Things Appeal More to Women (CS1 Appeal Perspective). In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) *(ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 145–156. https://doi.org/10.1145/3446871.3469761

[26] Dragan Mirković and S Lennart Johnsson. 2003. CODELAB: A Developers' Tool for Efficient Code Generation and Optimization. In *International Conference on Computational Science*. Springer, 729–738.

[27] Diba Mirza, Phillip T Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate teaching assistants in computer science: a systematic literature review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 31–40.

[28] Stephen Nutbrown and Colin Higgins. 2016. Static analysis of programming exercises: Fairness, usefulness and a method for application. *Computer Science Education* 26, 2-3 (2016), 104–128.

[29] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Trans. Comput. Educ.* 22, 3, Article 34 (jun 2022), 40 pages. https://doi.org/10.1145/3513140

[30] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education (TOCE)* 22, 3 (2022), 1–40.

[31] Laura Pappano. 2012. The Year of the MOOC. *The New York Times* 2, 12 (2012), 2012.

[32] Raymond Scott Pettit, John D Homer, Kayla Michelle McMurry, Nevan Simone, and Susan A Mengel. 2015. Are automated assessment tools helpful in programming courses?. In *2015 ASEE Annual Conference & Exposition*. 26–230.

[33] Vreda Pieterse and Janet Liebenberg. 2017. Automatic vs Manual Assessment of Programming Tasks. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '17)*. Association for Computing Machinery, New York, NY, USA, 193–194. https://doi.org/10.1145/3141880.3141912

[34] Emma Riese and Viggo Kann. 2020. Teaching assistants' experiences of tutoring and assessing in computer science education. In *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[35] Jonathan Sharples, P Blatchford, and R Webster. 2016. Making best use of teaching assistants. (2016).

[36] Janet Siegmund, Norbert Siegmund, and Sven Apel. 2015. Views on internal and external validity in empirical software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 9–19.

[37] SonarQube. 2023. SonarQube Documentation. https://docs.sonarqube.org/latest/. Accessed: 2023-01-13.

[38] Thomas Staubitz, Hauke Klement, Jan Renz, Ralf Teusner, and Christoph Meinel. 2015. Towards practical programming exercises and automated assessment in Massive Open Online Courses. In *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 23–30.

[39] Zahid Ullah, Adidah Lajis, Mona Jamjoom, Abdulrahman Altalhi, Abdullah Al-Ghamdi, and Farrukh Saleem. 2018. The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education* 26, 6 (2018), 2328–2341.

[40] Kurt VanLehn. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist* 46, 4 (2011), 197–221.

[41] Chris Wilcox. 2015. The role of automation in undergraduate computer science education. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 90–95.