Exploring Students' Self-Confidence in Their Programming Solutions

Sven Strickroth
sven.strickroth@ifi.lmu.de
LMU Munich
Munich, Germany

ABSTRACT

Learning programming is perceived as hard by many students. To support students, many e-assessment and intelligent tutoring systems have been developed. These systems can automatically evaluate student submissions and provide feedback. Despite comprehensive research on feedback modalities, little is known about students' confidence in the correctness of their submissions when requesting feedback. Also, educators hope that students get more confident and that automatically provided feedback helps students to improve and better self-assess their work. In this paper, first-semester students are asked about their confidence in passing a requested syntax or function test before the test results are revealed to them. Students can request feedback from each of the two provided test types twice in arbitrary order. The self-rated confidence, test outcomes, time needed to enter the confidence, and correlations are analyzed in detail. The results show that the majority of students has a high confidence in their submitted work. However, students frequently over-estimate the correctness and only few under-estimate it. There is a correlation between students' confidence in their submissions and their actual performance, but this cannot be used to make reliable predictions. The test pass rate for highly confident students is higher for syntax tests than for function tests and students need more time for entering their confidence for syntax than for function tests. Over the semester, the self-rated confidence decreases. When tests are reattempted, both correctness and self-assessment abilities show improvement.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

programming education, confidence, formative assessment, self-assessment, learning analytics

ACM Reference Format:
Sven Strickroth. 2024. Exploring Students' Self-Confidence in Their Programming Solutions. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653589

1 INTRODUCTION

Learning programming is an integral part of computer science education. For novices, however, programming is often perceived as hard [28]. Many competencies such as problem solving, computational thinking, and testing need to be mastered at the same time [12]. Learning programming requires practice and, therefore, exercises in which programming assignments are solved are usually an inherent part of programming courses. To support students, timely formative feedback is a driving factor for success [16]. Many e-assessment systems and Intelligent Tutoring Systems (ITS) have been developed to support students and educators [21, 36]. These systems can provide timely automatic generated feedback 24/7. The usage of such systems is sometimes referred to as (a variant of) self-assessment (e. g., [7, 10, 20]). Another variant of self-assessment is where students are asked to self-grade or self-correct their solution based on instructor defined criteria (e. g., [15]). In general, self-assessment enables students to get an insight on their own learning process and, in this context, to incrementally improve their solutions [7, 15]. Hence, self-assessment is a very desired skill.

There are, however, drawbacks of these automated systems. Students may misuse the feedback features or adopt unintended workflows if no special measures are taken: Students may apply undirected try-and-error strategies to pass the tests (cf. [5, 32]), use a system as an online compiler instead of an integrated development environment (IDE; e. g., [31]), or are not encouraged any more to test their programs on their own before submitting (cf. [13]).

An often overlooked aspect is students' confidence in their solutions. The aim of this research is to analyze student's confidence in their solution and their actual performance on programming assignments. Students are explicitly asked about their confidence in the correctness of their submission when they request automatically generated feedback before the result is shown to them. Hence, students are requested to reflect on their solution and receive immediate feedback on both the correctness of their submission against the specification and implicitly on their estimation of the correctness of their submission. The investigated research questions are:

- RQ1: How confident are students in the correctness of their submitted work?
- RQ2: How does the self-rated confidence correlate with the actual performance?
- RQ3: Are there differences over time, between two provided types of tests, failed and passed tests, or tests requested twice, and how long do students need to enter their confidence?

In this context, the self-rated confidence can be seen as the level of optimism students have in their performance after working on a particular task. Therefore, it is expected that this will implicitly include the student's assessment that they have adequately tested their solution, even if they were not given any specific criteria for testing other than the assignment specification.

In general, self-confidence is trust in one's own abilities and relates to "self-efficacy", a generalized concept defined by Bandura [6]. This concept describes an individual's confidence in their ability to take the actions required to achieve particular objectives [6]. Andrade [4] argues that formative self-assessments are a form of self-efficacy. Furthermore, being able to monitor one's own process and abilities, is an important factor for self-regulated learning [37].

The insights gained from answering these research questions are twofold: They contribute to the understanding of students' metacognitive skills and testing strategies in the context of learning programming. Furthermore, they can influence pedagogical strategies about when and how to address students' problems and help optimize e-assessment systems.

2 RELATED RESEARCH

Many automated e-assessment systems and ITS have been developed to support teaching [21, 25, 36]. Overall, many systems have shown to support students and their learning [3] – even very limited feedback can be helpful [35]. There are, however, differences in how these systems operate: ITS aim to be adaptive, mimic a human tutor, and provide personalized step by step help [3, 25] whereas e-assessment systems strive more towards to evaluate, test, or assess learner's competences, but can also provide formative feedback. There are two modes of operation: First, there are systems that automatically start the assessment and provide feedback to students immediately after uploading (e. g., JACK [14]). Second, there are submission systems that separate the upload from testing/feedback such as uploads (even invalid code) are possible until the deadline and then feedback is provided or feedback can be requested on-demand (e. g., GATE [35]). Particularly in the latter case, feedback capabilities can be easily limited to avoid abusing the system. Still, students might "delegate" the testing to the systems. It is unclear, how the test capabilities are used in such systems, i. e., at what stages students upload their work, or request feedback (e. g., when they got stuck, or want a feedback on the partial/final correctness, cf. [18]). Typically, existing systems provide feedback but do not require students to state their confidence in their solution.

Formative self-assessments in which students self-evaluate their work, have shown to have positive effects on the quality of the results [4, 15, 30]. Often specific criteria for evaluating a solution are given [4]. Haake et al. [15] investigated the self-assessment quality using teacher-provided criteria on open-ended questions in a computer science course. They found that assignments with a low or moderate extent are easier to self-assess and that very specific criteria can aid the assessment [15]. Alaoutinen and Smolander [2] developed a more generic questionnaire for students to self-assess their competences in programming courses. Their results indicate that students are quite accurate in assessing their knowledge and found a statistically significant correlation between the self-assessments and the exam results [2]. Strickroth [33] compared self-assessments in a programming course directly before and after doing peer reviews and found that the self-assessments get more accurate after reviewing fellow students' solutions [33]. A common issue is, however, that students often over or under-estimate their skills [2, 9, 15, 26]. Furthermore, there are contradictory results on whether students improve their accuracy over time or whether academically high performing students are better than low performing students (e. g., [9, 15, 26]). Providing detailed tests or checklists to students, also relieves them from thinking on the tests themselves. Furthermore, these self-assessments, however, do not provide any insights into the confidence of the students, and (by design) students do not receive detailed feedback on their solution.

(Self-)confidence was investigated in several studies: Layman et al. [24] found, on a course – not task – level, that students' self-confidence in their programming skills correlates with their overall performance. Ahsan et al. [1] investigated whether self-rated confidence is a predictor for the performance in programming comprehension tasks (multiple choice questions and a five-point confidence scale). The results indicate that the confidence levels are quite high for all students and there is no difference between high and low performing students. The confidence levels are only weakly correlated to performance [1]. The authors suggest that students should periodically assess their confidence levels so that they get an accurate view on their abilities. It remains unclear, however, whether or how the correctness of the self-rated confidence was fed back to the students in that study. A study on model comprehension tasks by Daun et al. [11] found that confidence was a good predictor for the correctness of the solutions. However, these studies may not be comparable to programming assignments. There also is research on factors contributing to self-efficacy in the context of CS education (e. g., [22]) but this is out of scope for this research.

Summing up, this research addresses the outlined research gaps on programming assignments. The ability to correctly assess one's own programming skills and performance is very important for learning, particularly by contrasting the internal assessment with external feedback [29]. Automatic e-assessment systems can support this, however, they may encourage students to "delegate" the assessment to the systems while self-assessments cannot help stuck students and take the task of "designing" tests from the students. The results on confidence being a predictor for performance and accuracy on self-assessments are inconclusive. This is addressed in this research by explicitly asking students for their confidence in their solution and mirroring back the actual correctness. Furthermore, the entered confidence is contrasted to the actual test outcome, and the time needed to enter the confidence, differences between test types as well as trends over time are investigated.

3 SETTING & METHODOLOGY

This study was conducted in the context of a first-semester introductory programming course (using Java) at LMU Munich, Germany in winter semester 2021/22. Approx. 900 students with computer science as a major or minor were enrolled in that course. Throughout the semester, 10 assignment sheets were distributed to the students. Eight of the assignment sheets contained at least one programming task. Working on the assignment sheets was voluntary. There also was no individual correction of students' solutions, but students could upload their solutions to the e-assessment system GATE [34, 35] and request automatically generated feedback. Additionally, solutions and questions were discussed in exercise sessions held by student teaching assistants in the week after the deadline.

GATE is designed as a submission system where students can upload new solution attempts without any restrictions until the deadline is reached. After the initial submission, syntax and (black box) function tests can be offered to the students. Each configured test can be requested by the students in arbitrary order. GATE allows educators to limit the number of times a student can request a test to prevent system abuse and to encourage the students to test their submissions themselves, plan when to use the offered tests, and not to fully rely on the system.

In the course, two tests were offered for every assignment (if possible), one syntax test and one function test which could be requested at most twice each. Before presenting the test results to the students, they were asked "How confident are you that the test will pass?". An HTML slider with values from 1 to 100 was used that could be freely positioned. The ends were labeled with "very uncertain" and "very certain". The default value was 50. If students did not interact with the slider, an alert box was shown that kindly asked to actually rate their confidence. Still, it was possible that students did not interact with the slider and just submitted the form to see their test results. These cases will not be considered in the detail analyses. For the analysis, 5 assignments from 5 different weeks are considered (cf. Table 1). One syntax test and one function test were offered for each task. The other assignments are not comparable (e. g., no function tests possible due to design freedoms).

The dataset contains all (intermediate) submissions of the students and the requested tests including meta data such as the timestamp, the test result, the entered confidence, an indicator whether the students were reminded to enter their confidence, and the time needed to proceed to the results page (i. e., enter their confidence). In the following results section, the self-rated confidence, correlations with the actual pass rate, time needed to enter the confidence are investigated across all tasks, but also differences between the two types of tests, failed and passed tests and over time are explored.

Non-parametric tests are used in the analysis because they do not require e. g. normal distributed data (cf. [8, 19]). Typically, these tests are stricter than their parametric counterparts [19]. For binary (dichotomous) data and dependent samples, the McNemar test is used as the significance test (cf. [8, 19]). For non dichotomous data, the Mann-Whitney UTest (independent) and the Wilcoxon rank sign test (dependent samples) are used.

All students were explicitly asked whether they voluntarily agree that their anonymized submissions and log data can be used for research. Only students who agreed were asked for their confidence. Students were able to give consent throughout the whole semester, so a small number of students joined late. Data from 770 submissions by 371 different students are included in the dataset.

4 RESULTS

Overall, 2023 tests were requested by the students (952 syntax tests and 1071 function tests). In 92 % of the tests, the students entered a self-rated confidence ($n = 1867$) – in 93 % of these cases ($n = 1700$) without being reminded to do so.

4.1 Analysis of the Self-Rated Confidence

Figure 1 shows a (logarithmic) histogram of all the confidence values entered; most of the values are for the extreme values 100, 1, and 50. In the following, the 156 cases (71 syntax and 85 function tests) where no confidence was entered are excluded. The median (m) self-rated confidence is 100, and the arithmetic mean ($\bar{x}$) is 84.

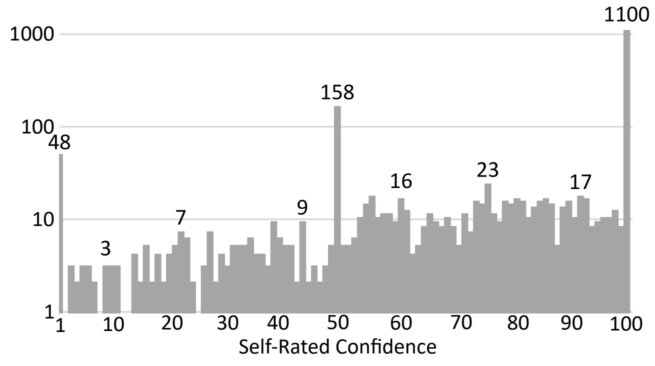

Figure 1: Histogram of all confidence values entered

Students requested about 100 more function tests than syntax tests (cf. lines 2–3 in Table 1). Although the median self-rated confidence is 100 for both test types, there is a statistically significant difference between syntax $\bar{x} = 87$ and function tests $\bar{x} = 81$ (UTest: $Z = 11.304$, $p < .001$) with a moderate effect size ($r = .25$, cf. [27]).

Over the semester, the entered self-rated confidences decrease (statistically significant with a small negative correlation of the confidence with the week number in the semester, Spearman's $\rho(1865) = -.154$, $p < .001$). The maximum confidence entered is 100 for all tasks. For the first four tasks, the minimum confidence entered is 1 and the median is 100. For the fifth task, the minimum is 55 for the syntax test and 19 for the function test, and the overall median is 97. The last 4 lines in Table 1 show no difference in the median and mean confidence for the syntax tests of task 1 and task 5. However, the median and the mean confidence are lower for the fifth task. This difference is statistically significant (UTest: $Z = -4.668, p < .001$) with a moderate effect size of $r = .25$.

4.2 Test Result and the Self-Rated Confidence

The majority of the requested tests passed (69 %); 89 % of the syntax and 51 % of the function tests (cf. lines 1–3 in Table 1). This difference between the two types of tests is statistically significant (UTest: $Z = -17.413$, $p = .001$) with a moderate to large effect size of $r = .4$.

Overall (line 1 in Table 1), the majority of students was highly confident that they would pass the requested test. However, the "$C = 100$" column shows that only 78 % of the requested tests passed for students who were most confident; 93 % for syntax tests and 61 % for function tests. Contrast this with the "$C = 1$" column where about 2 % of the students entered the lowest possible confidence but passed the tests in 34 % of the cases overall, 64 % of the cases for the syntax tests, and only 8 % of the cases for the function tests.

Looking from the "other side", lines 4–5 in Table 1 show that across all tests that passed, 86 % of the students entered a high confidence and 66 % entered the highest confidence. Only 8 % of the students entered a low confidence. Across all failed tests, 69 % of the students entered a high confidence, 42 % entered the highest confidence, and only 16 % a low confidence. Overall, the self-rated confidence is statistically significantly higher for tests that actually

Table 1: Overview of the tasks, test types, and self-rated confidence (T: median/mean time needed for entering the confidence, C: median/mean confidence, R: test pass ratio; last columns: *n* cases in a range with a specific confidence and test pass ratio R)

Aspect	n	T		C		R	$C = 1$		$C = 100$		$C < 33$		$33 \leq C \leq 66$		$C > 66$	
		m	$\bar{x}$	**m**	$\bar{x}$		**n**	**R**	**n**	**R**	**n**	**R**	**n**	**R**	**n**	**R**
All Tasks	1867	4	4.5	100	84	.69	48	.34	1100	.78	142	.36	147	.37	1491	.74
Only Syntax Tests	881	4	5.1	100	87	.89	22	.64	572	.93	57	.63	60	.65	733	.91
Only Function Tests	986	3	3.9	100	81	.51	26	.08	528	.61	85	.18	87	.18	758	.57
All Passed Tests	1284	3	4.6	100	89	1	16	1	853	1	51	1	55	1	1101	1
All Failed Tests	583	3	4.3	86	73	.00	32	.00	247	.00	91	.00	92	.00	390	.00
Task 1: Nested loops	616	3	5.8	100	91	.85	16	.50	448	.90	29	.48	30	.50	556	.87
Task 2: Reverse Array content	453	3	4.4	100	78	.60	15	.20	230	.69	53	.34	55	.36	314	.66
Task 3: Abstr. classes & polymorph.	389	3	3.8	100	81	.49	14	.29	208	.56	32	.22	33	.21	295	.53
Task 4: Recursion	318	3	3.4	100	83	.75	3	.34	172	.83	26	.46	27	.48	254	.80
Task 5: Using an external library	91	3	3.1	97	85	.69	0	–	42	.76	2	.00	17	.76	72	.69
Task 1 Only Syntax Tests	295	6	7.2	100	90	.91	10	.70	220	.95	16	.69	15	.67	264	.93
Task 1 Only Function Tests	321	3	4.5	100	91	.79	6	.17	228	.86	13	.23	16	.88	292	.82
Task 5 Only Syntax Tests	42	3	3.2	100	90	.98	0	–	23	.96	0	–	7	1	35	.97
Task 5 Only Function Tests	49	3	3.0	85	81	.45	0	–	19	.53	2	.00	10	.56	37	.43

passed ($m = 100$, $\bar{x} = 89$) than for those that failed ($m = 86$, $\bar{x} = 73$; UTest: $Z = 11.304$, $p < .001$) with a moderate effect size of $r = .26$.

Over the semester, the test pass rates decrease (statistically significant with a small negative correlation with the week number in the semester, Spearman's $\rho(1865) = -.152$, $p < .001$).

The last six columns in Table 1 show the pass rates for different confidence ranges. In the following, self-rated confidences below 33 are referred to as *low*, between 33 and 66 as *middle*, and above 66 as *high*. Over all tasks and for the syntax as well as the function tests, the number of students choosing a low or middle confidence is about the same (≈ 6 to 8 %), except for task 5 where 2 % and 19 % entered a low respective middle confidence. Also, the pass rates are always about the same for both ranges or higher for the middle range, except for task 3 and the detail analysis of the function test of task 1. Notable are the high pass rates for the syntax tests for both of these ranges (63 and 65 %), and the same low pass rates of 18 % for the function tests. Salient is task 5 where the pass rate for the medium range is significantly higher than for both the low confident and the high confident students. Finally, the pass rates are always significantly higher for the high confident students than for the low and middle confident students, except for task 5, and the detail analyses of function tests of task 1 and 5. When comparing the pass rates of the three classes (line 1 in Table 1), all are statistically significantly different (UTest, $p < .001$) with small effect sizes (low vs. middle: $r = .20$, middle vs. high: $r = .13$, low vs. high: $r = .23$).

Finally, the syntax and function tests are compared individually for the first and last task (last four lines in Table 1). The pass rate for the syntax test was already quite high at 91 % for the first task and increased to 98 %. Also, the test pass rate for medium and highly confident students increased and the number of less confident students decreased. The picture is different however for the function tests: Most students are still very confident to pass the tests, but the overall pass rate decreases from 79 to 45 % as well as the pass rate for highly confident students decreases from 82 to 43 %. In all two cases where students entered a low confidence, the tests failed.

Overall, there is a positive correlation between the confidence and the actual test result with a moderate effect size (cf. Table 2). The correlation is slighter stronger for the function test. Over the semester, correlations with small to moderate effect sizes could be found for all tasks except for the last task where it is not significant. The effect size seems to be strongest for the first task and to decrease over the semester.

Table 2: Spearman correlations between the confidence and passing a test (ES: effect size: small or moderate, cf. [27])

Aspect	n	ρ	p	ES
All Tasks	1867	.262	<.001	m
Only Syntax Tests	881	.211	<.001	s
Only Function Tests	986	.257	<.001	m
Task 1	616	.254	<.001	m
Task 2	453	.242	<.001	m
Task 3	389	.191	<.001	s
Task 4	318	.231	<.001	s
Task 5	91	.189	.073	(s)
Task 1 Only Syntax Tests	295	.257	<.001	m
Task 1 Only Function Tests	321	.257	<.001	m
Task 5 Only Syntax Tests	42	-.134	.40	(s)
Task 5 Only Function Tests	49	.141	.34	(s)

4.3 Analysis of Tests Requested Twice

In 513 cases, students requested the same type of test for an assignment twice (cf. Table 3). No confidence for both tests was entered in 21 cases, only for the first test in 18, and only for the last test in 22 cases – these are excluded. Hence, 452 cases are analyzed here.

Overall, most of the first tests failed ($\bar{x} = .33$) and most of the second tests passed ($\bar{x} = .71$). This difference is statistically significant (McNemar: $p < .001$) with a large effect size (Cohen's $\omega = .426$, cf. [27]). The majority of students could fix their error(s). There

Table 3: Comparison of tests executed twice on an assignment by the very same student (for the legend see Table 1)

Test	n	T		C		R	$C = 1$		$C = 100$		$C < 33$		$33 \leq C \leq 66$		$C > 66$	
		m	$\bar{x}$	m	$\bar{x}$		n	R	n	R	n	R	n	R	n	R
All 1st	452	3	4.2	100	81	.33	14	.29	233	.40	45	.24	65	.32	342	.35
All 2nd	452	3	3.5	100	85	.71	9	.22	274	.79	30	.30	62	.53	360	.77
Syntax 1st	188	4	5.1	100	83	.71	5	.60	104	.79	18	.56	26	.81	144	.72
Syntax 2nd	188	3	3.1	100	89	.91	3	.67	133	.92	10	.70	15	.93	163	.93
Function 1st	264	3	3.6	96.5	79	.06	9	.11	129	.09	27	.04	39	.00	198	.08
Function 2nd	264	3	3.8	100	82	.56	6	.00	141	.67	20	.10	47	.40	197	.64

also is a statistically significant difference between the first entered confidence ($m = 93$, $\bar{x} = 77$, [1,100]) and second entered confidence ($m = 100$, $\bar{x} = 82$, [1,100]; Wilcoxon sign rank test, $Z = -3.481$, $p < .001$) with a small effect size of $r = .16$. The percentage of highly confident students who failed the second test decreased (still 23 %), but the percentage of low confident students who passed it increased (from 24 % to 30 %). For both tests, there are statistically significant correlations between the entered confidence and the pass rate with a larger effect size for the second test (cf. Table 4).

Table 4: Spearman correlations between the entered confidence and passing a test for tests requested twice (ES: effect size: very small, small or moderate, cf. [27])

Test	n	ρ	p	ES
All 1st tests	452	.152	.001	s
All 2nd tests	452	.272	<.001	m
Syntax 1st tests	188	.168	.021	s
Syntax 2nd tests	188	.037	.613	(vs)
Function 1st tests	264	.129	.036	s
Function 2nd tests	264	.301	<.001	m

When analysing the two types of tests individually, most first syntax tests (71 %) and nearly all second tests (91 %) passed. 68 students increased their confidence, and only 22 decreased it of which only 2 could not fix their error between the two tests. Only for the first syntax test, but not for the second, there is a statistically significant correlation between confidence and pass rate with a small effect size (cf. Table 4). For the function tests it is noticeable that nearly all first tests failed (about 94 %) and for the second test still 44 % failed. No new errors were introduced after the first test passed. There are more students ($n = 90$) who have increased their confidence than students ($n = 68$) who have decreased it. For both function tests, there is a statistically significant correlation between confidence and pass rate. The effect size is small for the first test but moderate for the second test (cf. Table 4). The differences in the test result between the first and second test are statistically significant for both types of tests (McNemar: both $p < .001$), but the effect size is larger for the syntax test (Cohen's $\omega = .234$ vs. $\omega = .354$). However, the difference in the self-rated confidence between the first and second test is only statistically significant for the syntax tests (Wilcoxon sign rank test, $Z = -4.162$, $p < .001$) with a moderate effect size of $r = .3$ and not for the function tests ($Z = -1.363$, $p = .173$).

4.4 Time Needed to Enter the Confidence

Overall, the median time needed for entering the confidence is 4 s ($\bar{x} = 4.5$ s). The time needed decreases over the semester. Still, there are statistically significant differences on the time needed between the syntax $\bar{x} = 5.1$ and function $\bar{x} = 3.9$ test (UTest: $Z = -10.285$, $p < .001$) with a moderate effect size of $r = .24$. Interestingly, it also took the students minimal longer to enter their confidence for tests that passed $\bar{x} = 4.6$ than for tests that failed $\bar{x} = 4.3$ (UTest: $Z = 2.840$, $p = .005$, very small effect size of $r = .07$). While there is a very small positive correlation between the time needed and passing a test (Spearman's $\rho(1865) = .066$, $p = .004$), there is a negative correlation between the time needed and the self-rated confidence ($\rho(1865) = -.279$, $p < .001$) with a moderate effect size.

When comparing the syntax tests and the function tests requested twice, there is a statistically significant difference between the time needed for entering the first ($m = 3$ s, $\bar{x} = 4.5$ s) and second confidence ($m = 3$ s, $\bar{x} = 3.7$ s; UTest: $Z = -5.858$, $p < .001$) with a moderate effect size of $r = .28$.

5 DISCUSSION

The results show that most students entered a high confidence in their work – even 59 % of the students entered the highest confidence – and only about 8 % entered a low confidence (RQ1). This suggests that most students expect to pass the tests and the majority of students do not seem to request a test for a known incomplete solution e. g. to get to know the test cases.

Overall, there is a moderate positive correlation between the self-rated confidence and passing a test (RQ2). However, about one-third of the high-confidence students over-estimate the correctness of their work (about 21 % of all students) and about one-third of the low-confidence students under-estimate the correctness of their work (about 3 % of all students). This is in line with related research (e. g., [1, 2, 9, 15, 26, 33]). Similar results with an accuracy of about 60–75 % have also been reported for self- and peer-assessments (e. g., [17, 33]). One reason could be the Dunning-Kruger effect [23], which states that less competent people over-estimate their competence without being aware of it. Narciss [29] argues that it is particularly important to explicitly contrast internal assessment with external feedback as done in this study. This encourages students to reflect on their perceived confidence and actual performance and should be conducted regularly. Since there is only a moderate correlation between self-rated confidence and actual performance, it cannot be used directly for predictions. This is consistent with a study on program comprehension [1]. Nevertheless, correct submissions receive a higher self-rated confidence than incorrect ones.

Now differences over time, for tests requested twice, and between the two types of tests are discussed (RQ3). The decreasing number of submissions over the semester is normal. At the same time, the self-rated confidence decreased slightly over the semester. This may reflect the experienced complexity of the assignments or indicates students' growing awareness of possible edge cases in the assignments. Also, the pass rates for high-confidence students decreased. This was contradictory to the expectation of the author. One reason may be that the assignments got more complex quicker than (test) competencies were developed. The decreasing ratio of passed tests supports this. Still, between two requested tests of the same type, most students were able to correct their submission and the pass ratio for high confident students improved. This result is not surprising, but it is good to see that the empirical data supports this assumption. However, the pass rate for low confident students also increased. This is counter intuitive and may be a way to (better) identify students needing special support. Overall, the the pass rates for high confident students are higher for syntax than for function tests and increase less between tests requested twice. One reason for this may be that after the first lectures not much new syntax is introduced and students become more familiar with an IDE. Task 3 supports this assumption, as the introduction of inheritance with its associated syntax decreased the syntax pass rate once for this task. Towards the end of the semester, failing syntax tests of high-confidence students underline the need for special support for these students. Hence, asking for self-rated confidence may help to identify these students. Interestingly, syntax tests are requested at a high rate throughout the whole semester, even though students are expected to use an IDE (also reported in [35]). Function tests may fail more often (especially on the test requested first), because students do not yet know all the test/edge cases. This could explain the massive increase in the pass rate for high-confidence students, however, the pass rate of only 43 % is surprising for the last task.

Andrade [4] argues that one should not only focus on accuracy for self-assessments. This is especially true in this research. Is it enough to be able to correctly predict the test result in advance, i. e., to have low confidence and actually fail the test, but not being able to correct the error? Unexpected failures (i. e., high confidence but failing a test) seem to be mostly caused by "unexpected" edge cases or minor errors and were fixable for most students, while expected failures may be mostly caused by lack of knowledge or misconceptions. Hence, educators should pay attention to identify such students. Fortunately, only about 8 % of the students entered a low confidence. Further research is needed, including feedback strategies that consider an (over or under-estimated) self-rated confidence in order to better address the needs of the learners.

Most students entered their confidence within 3–6 seconds. This indicates that students did not take or need much time to think about their confidence once they had decided to request a test. The decrease in time over the semester and for the second test of tests requested twice is expected as students become more familiar with the whole process. However, it is noteworthy and surprising that students need (slightly) more time to enter their confidence for both the syntax tests and the tests that actually passed. Especially for syntax tests, the result should already be known by using an IDE.

6 LIMITATIONS & THREATS TO VALIDITY

The analysis is based on a single course in one university in a specific setting where tests could be requested maximum twice. Also, some student may have cooperated to get more than 4 tests. There are two levels of self-selection: First, only students who voluntary submitted solutions on the systems are included in the dataset. Hence, there might be a bias towards for more engaged students. There could, however, also be a bias in the opposite direction that high performing students did not submit their solutions for as "too easy" perceived tasks. Secondly, only students who have explicitly consented to the use of their data were considered (≈ 80 % agreed).

The HTML slider may be seen as not optimal, since a default value was given. However, the slider was chosen to allow students to freely choose a confidence and not to use pre-defined ranges in order to be able to investigate chosen values and smaller changes.

Only a quantitative analysis was conducted. The function tests may be too strict in some cases and may even fail for minor errors such as typos in the output. This may lower the level of over-estimation but it should not have a significant impact on the results as multiple assignments are analyzed and a wrong output which does not fulfill the assignment specification is strictly speaking not fully correct. For ruling this completely out, an in-depth qualitative analysis of the submissions and test results would be necessary.

7 CONCLUSIONS AND OUTLOOK

Students were asked by the e-assessment system GATE about their confidence in passing a requested test before its result was revealed to them. The results show that students have a high confidence in their submissions and often over-estimate their correctness. Also, few students under-estimate their correctness. There is a moderate correlation between the self-rated confidence and the actual performance but this cannot be used for reliable predictions. The test pass rate for highly confident students is higher for syntax tests than for function tests and students need more time for entering their confidence for syntax than for function tests. Over the semester, the self-rated confidence decreases. When tests are reattempted, both correctness and self-assessment abilities seem to improve.

In this study, only a quantitative evaluation was conducted which cannot reveal reasons for choosing a high or low confidence. Further research should investigate how the self-rated confidence, test behavior, and final correctness of the submissions correlate. In addition, it would be interesting to investigate feedback strategies based on the self-rated confidence and the outcome of the tests to better identify and help struggling students. Here, different feedback may be provided depending on high/low confidence and passing or failing a test, e. g. providing code optimizations in the first case and more basic feedback or additional tasks in the latter case. Furthermore, it might be interesting to see whether there are differences when the confidence is entered after seeing (some of) the test cases. Combining criteria-led self-assessments with self-rated confidence could encourage students to make an explicit estimation before knowing the criteria. In the context of learning analytics, self-rated confidence and actual test scores could be used to highlight knowledge gaps and misconceptions for educators, or to visualize test and confidence history for learners as a (meta-)cognitive aid.

REFERENCES

[1] Zubair Ahsan, Unaizah Obaidellah, and Mahmoud Danaee. 2022. Is Self-Rated Confidence a Predictor for Performance in Programming Comprehension Tasks? *APSIPA Transactions on Signal and Information Processing* 11, 1 (2022). https://doi.org/10.1561/116.00000041

[2] Satu Alaoutinen and Kari Smolander. 2010. Student self-assessment in a programming course using bloom's revised taxonomy. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. ACM. https://doi.org/10.1145/1822090.1822135

[3] Ali Alkhatlan and Jugal Kalita. 2019. Intelligent Tutoring Systems: A Comprehensive Historical Survey with Recent Developments. *International Journal of Computer Applications* 181, 43 (mar 2019), 1–20. https://doi.org/10.5120/ijca2019918451

[4] Heidi L. Andrade. 2019. A Critical Review of Research on Student Self-Assessment. *Frontiers in Education* 4 (aug 2019). https://doi.org/10.3389/feduc.2019.00087

[5] Ryan Baker, Jason Walonoski, Neil Heffernan, Ido Roll, Albert Corbett, and Kenneth Koedinger. 2008. Why students engage in "gaming the system" behavior in interactive learning environments. *Journal of Interactive Learning Research* 19, 2 (2008), 185–224.

[6] Albert Bandura. 1986. Social foundations of thought and action. *Englewood Cliffs, NJ* 1986, 23-28 (1986).

[7] Bruno Baruque and Álvaro Herrero. 2015. Self-Assessment Web Tool for Java Programming. In *Advances in Intelligent Systems and Computing*. Springer International Publishing, 583–592. https://doi.org/10.1007/978-3-319-19713-5_51

[8] Ralf Bender, Stefan Lange, and Andreas Ziegler. 2007. Wichtige Signifikanztests. *Deutsche Medizinische Wochenschrift* 132 (2007), e24–e25. https://doi.org/10.1055/s-2007-959034

[9] David Boud, Romy Lawson, and Darrall G. Thompson. 2014. The calibration of student judgement through self-assessment: disruptive effects of assessment patterns. *Higher Education Research & Development* 34, 1 (nov 2014), 45–59. https://doi.org/10.1080/07294360.2014.934328

[10] Peter Brusilovsky and Sergey Sosnovsky. 2005. Individualized exercises for self-assessment of programming knowledge. *Journal on Educational Resources in Computing* 5, 3 (sep 2005), 6. https://doi.org/10.1145/1163405.1163411

[11] Marian Daun, Jennifer Brings, Patricia Aluko Obe, and Viktoria Stenkova. 2021. Reliability of self-rated experience and confidence as predictors for students' performance in software engineering. *Empirical Software Engineering* 26, 4 (jun 2021). https://doi.org/10.1007/s10664-021-09972-6

[12] Michael Ebert and Markus Ring. 2016. A presentation framework for programming in programing lectures. In *Proc. EDUCON*. IEEE, 369–374.

[13] Stephen H. Edwards. 2003. Improving student performance by evaluating how well students test their own programs. *Journal on Educational Resources in Computing* 3, 3 (sep 2003), 1. https://doi.org/10.1145/1029994.1029995

[14] Michael Goedicke and Michael Striewe. 2009. A Flexible and Modular Software Architecture for Computer Aided Assessments and Automated Marking. In *Proceedings of the First International Conference on Computer Supported Education*. SciTePress - Science and and Technology Publications. https://doi.org/10.5220/0001966900540061

[15] Joerg M. Haake, Niels Seidel, Marc Burchart, Heike Karolyi, and Regina Kasakowskij. 2021. Accuracy of self-assessments in higher education. In *DELFI 2021*. Gesellschaft für Informatik e.V., Bonn, 97–108.

[16] John Hattie and Helen Timperley. 2007. The Power of Feedback. *Review of Educational Research* 77, 1 (mar 2007), 81–112. https://doi.org/10.3102/003465430298487

[17] Niels Heller and Francois Bry. 2019. Organizing Peer Correction in Tertiary STEM Education: An Approach and its Evaluation. *International Journal of Engineering Pedagogy (iJEP)* 9, 4 (2019), 16–32. https://doi.org/10.3991/ijep.v9i4.10201

[18] Johan Jeuring, Hieke Keuning, Samiha Marwan, Dennis Bouvier, Cruz Izu, Natalie Kiesler, Teemu Lehtinen, Dominic Lohr, Andrew Peterson, and Sami Sarsa. 2022. Towards Giving Timely Formative Feedback and Hints to Novice Programmers. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM. https://doi.org/10.1145/3571785.3574124

[19] Amandeep Kaur and Robin Kumar. 2015. Comparative analysis of parametric and non-parametric tests. *Journal of computer and mathematical sciences* 6, 6 (2015), 336–342.

[20] Judy Kay, Lichao Li, and Alan Fekete. 2007. Learner Reflection in Student Self-Assessment. In *Proceedings of the Ninth Australasian Conference on Computing Education - Volume 66* (Ballarat, Victoria, Australia) *(ACE '07)*. Australian Computer Society, Inc., AUS, 89–95.

[21] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education* 19, 1, Article 3 (Sept. 2018), 43 pages. https://doi.org/10.1145/3231711

[22] Attila Kovari and Jozsef Katona. 2023. Effect of software development course on programming self-efficacy. *Education and Information Technologies* (feb 2023). https://doi.org/10.1007/s10639-023-11617-8

[23] Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology* 77, 6 (1999), 1121–1134. https://doi.org/10.1037/0022-3514.77.6.1121

[24] L. Layman, L. Williams, J. Osborne, S. Berenson, K. Slaten, and M. Vouk. 2005. How and Why Collaborative Software Development Impacts the Software Engineering Course. In *Proceedings Frontiers in Education 35th Annual Conference*. IEEE. https://doi.org/10.1109/fie.2005.1611964

[25] Nguyen-Thinh Le, Sven Strickroth, Sebastian Gross, and Niels Pinkwart. 2013. A Review of AI-Supported Tutoring Approaches for Learning Programming. In *Advanced Computational Methods for Knowledge Engineering - Proceedings of the 1st International Conference on Computer Science, Applied Mathematics and Applications (ICCSAMA) (Studies in Computational Intelligence, 479)*. Springer Verlag, Berlin, Germany, 267–279. https://doi.org/10.1007/978-3-319-00293-4_20

[26] Magdeleine D.N. Lew, W.A.M. Alwis, and Henk G. Schmidt. 2010. Accuracy of students' self-assessment and their beliefs about its utility. *Assessment & Evaluation in Higher Education* 35, 2 (mar 2010), 135–156. https://doi.org/10.1080/02602930802687737

[27] Andrey Lovakov and Elena R. Agadullina. 2021. Empirically derived guidelines for effect size interpretation in social psychology. *European Journal of Social Psychology* 51, 3 (April 2021), 485–504. https://doi.org/10.1002/ejsp.2752

[28] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proc. ITiCSE*. 55–106. https://doi.org/10.1145/3293881.3295779

[29] Susanne Narciss. 2008. Feedback strategies for interactive learning tasks. *Handbook of research on educational communications and technology* 3 (2008), 125–144.

[30] Zane Olina and Howard J. Sullivan. 2004. Student self-evaluation, teacher evaluation, and learner performance. *Educational Technology Research and Development* 52, 3 (sep 2004), 5–22. https://doi.org/10.1007/bf02504672

[31] Sven Eric Panitz and Ralf Dörner. 2018. Nicht nur Bestehen, sondern auch Verstehen: Ein Werkzeug für direktes, kontinuierliches Feedback beim Lernen von Programmieren. In *Proceedings der Pre-Conference-Workshops der 16. E-Learning Fachtagung Informatik co-located with 16th e-Learning Conference of the German Computer Society (DeLFI 2018) (CEUR Workshop Proceedings, Vol. 2250)*. CEUR-WS.org.

[32] Raymond Pettit, John Homer, Roger Gee, Susan Mengel, and Adam Starbuck. 2015. An Empirical Study of Iterative Improvement in Programming Assignments. In *Proc. SIGCSE*. ACM, 410–415. https://doi.org/10.1145/2676723.2677279

[33] Sven Strickroth. 2023. Does Peer Code Review Change My Mind on My Submission?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023)*. Association for Computing Machinery, 498–504. https://doi.org/10.1145/3587102.3588802

[34] Sven Strickroth and Florian Holzinger. 2022. Supporting the Semi-Automatic Feedback Provisioning on Programming Assignments. In *Methodologies and Intelligent Systems for Technology Enhanced Learning, 12th International Conference*. Springer International Publishing, Cham, 13–19. https://doi.org/10.1007/978-3-031-20617-7_3

[35] Sven Strickroth, Hannes Olivier, and Niels Pinkwart. 2011. Das GATE-System: Qualitätssteigerung durch Selbsttests für Studenten bei der Onlineabgabe von Übungsaufgaben?. In *Tagungsband der 9. e-Learning Fachtagung Informatik (DeLFI)*. Gesellschaft für Informatik e.V., Bonn, Germany, 115–126. https://dl.gi.de/handle/20.500.12116/4740

[36] Sven Strickroth and Michael Striewe. 2022. Building a Corpus of Task-based Grading and Feedback Systems for Learning and Teaching Programming. *International Journal of Engineering Pedagogy (iJEP)* 12, 5 (Nov. 2022), 26–41. https://doi.org/10.3991/ijep.v12i5.31283

[37] Barry J. Zimmerman. 2000. Self-Efficacy: An Essential Motive to Learn. *Contemporary Educational Psychology* 25, 1 (jan 2000), 82–91. https://doi.org/10.1006/ceps.1999.1016