Are automated assessment tools helpful in programming courses?

Mr. Raymond Scott Pettit, Abilene Christian University

Raymond S. Pettit teaches courses in programming, artificial intelligence, objected oriented design, algorithms, theory of computation, and related subjects in ACU’s School of Information Technology and Computing. Prior to joining the ACU faculty, he spent twenty years in software development, research, and training the Air Force Research Lab and NASA’s Langley Research Center as well as private industry. His current research focuses on how automated assessment tools interact with student learning in university programming courses.

John D Homer, Abilene Christian University

John Homer, Abilene Christian University

Dr. John Homer is an associate professor in the School of Information Technology and Computing at Abilene Christian University. His current research focuses on risk assessment, computational theory, and automated assessment tools.

Ms. Kayla Michelle Holcomb, Abilene Christian University

Kayla Holcomb, Abilene Christian University

Ms. Kayla Holcomb is an undergraduate research assistant majoring in computer science in the School of Information Technology and Computing at Abilene Christian University. She is currently assisting with research in computer science education and digital assessment tools.

Nevan Simone, Abilene Christian University

Nevan Simone, Abilene Christian University

Mr. Nevan Simone is an undergraduate research assistant majoring in computer science in the School of Information Technology and Computing at Abilene Christian University. He is currently assisting with research in computer science education and digital assessment tools.

Dr. Susan A. Mengel, Texas Tech University

Susan Mengel, Texas Tech University

Dr. Susan Mengel is an associate professor in the Computer Science Department of the Edward E. Whitacre Jr. College of Engineering at Texas Tech University. Her research interests include computer science education, computer security, and information retrieval.

©American Society for Engineering Education, 2015
Are Automated Assessment Tools Helpful in Programming Courses?

Abstract

Automated assessment tools (AATs) are growing in popularity in introductory programming courses, but researchers may have a difficult time synthesizing valid data to draw conclusions about the tools’ usefulness. Our first step addressing this issue was to break down our overriding question—are automated assessment tools helpful in programming courses?—into four more specific questions: (1) Have AATs proven to be helpful in improving student learning? (2) Do students think that AATs have improved their performance? (3) After having used the tools, do instructors think that the tools have improved their teaching experiences? and (4) Is the assessment performed by AATs accurate enough to be helpful? In discussing the many AATs that exist, many researchers have only reported results relevant to one or two of these specific questions. We address each of our four questions separately and draw on data from 24 different tools to arrive at our conclusions. We determine that the literature demonstrates AATs helpfulness in student learning, instructor support, and assessment accuracy. However, we found results about students’ opinions regarding the helpfulness of AATs to be inconclusive. Given our findings, we make suggestions both for instructors using these tools and to researchers creating them.

1. Introduction

As their popularity has grown, automated assessment tools (AATs) have become a staple in computer science courses. Many instructors and researchers have developed their own tools in order to advance certain features for their courses. These instructors assume that these tools will help their courses in some way, but may have a difficult time synthesizing valid data to draw conclusions about the tools’ usefulness and to make related decisions about their own tool development. In this paper, we started out with a basic research question: are automated assessment tools helpful in programming courses?

To examine this question, we looked at four distinct areas that could be measured about AATs, then looked at published research results about AATs and categorized those results according to our measurement areas: (1) evidence of learning advancement in students, (2) students’ impressions and attitudes toward the tools, (2) instructors’ impressions and satisfaction with the tools, and (4) measures of accuracy of the tools in assessing assignments. We formulated each of these four areas into corresponding research questions:

1) Have AATs proven to be helpful in improving student learning?
2) Do students think that AATs have improved their performance?
3) After having used the tools, do instructors think that the tools have improved their teaching experiences?
4) Is the assessment performed by AATs accurate enough to be helpful?

In Section 2, we review related works and list the tools that we reference later in the paper. Section 3 explains our approach to analyzing the body of research concerning AATs and
provides an explanation of what it means for an AAT to be “helpful” for each of the four research questions stated above. Section 4 constitutes the bulk of this paper, containing our analysis of relevant publications. We address each research question with results synthesized from meaningful papers. Section 5 includes a summary conclusion and proposes future work to help improve AATs in the future.

2. Related works

Researchers have introduced and evaluated many automated assessment tools in individual papers. Most AATs accept and run student-provided source code, but some assess code tracing, promote good style techniques, or encourage a test-driven approach. In our review of existing literature, we found data relevant to our research questions for these tools: Aari,1 Athene,2 Autograde,3 AutoLep,4 Automatic Marker,5 BlueFix,6 BOSS,7 CAP,8 Ceilidh,9 COALA,10 CodeWrite,11 CourseMarker,12 Curator,13 JUG,14 Mooshak,15 PGSE,16 Retina,17 Style++,18 Submit,19 Test My Code,20 TRAKLA,21 TRAKLA2,22 Trillium,23 and Web-CAT.24 We recognize that published research exists on many comparable tools, but we did not readily find data that addressed our questions. Still, we will continue to track results relevant to other tools, including: ALOHA,25 ASAP,26 ASSYST,27 AutoGradeMe,28 AWAT,29 Bottlenose,30 Clockit,31 CloudCoder,32 CodeAssessor,33 CodeLab,34 CTPracticals,35 EasyAccept,36 EduComponents,37 ELP,38 FitchFork,39 HoGG,40 JEWL,41 Junit,42 Kattis,43 Linuxgym,44 Marmoset,45 Moe,46 Online Judge,47 PASS,48 Peach,49 ProgTest,50 ProtoAPOGEE,51 Quiver,52 QuizPACK,53 Resolver,54 RoboCode,55 RoboProf,56 Scheme-robo,57 TRY,58 USACO’s,59 UVA Online Judge,60 VERKKOKE,61 WebToTeach,62 and WeBWorK-JAG.63

Several excellent surveys exist on automated assessment tools and their behaviors. In 2005, Ala-Mutka evaluated known tools at a time when there were few methodical studies over the effects of automated assessment and feedback on students’ learning.64 Also in 2005, Douce provided an outline of the development history of AATs and a survey of related research.65 He proposed criteria for evaluating these tools, including whether the system “does what it is supposed to do,” “is liked by its users” (instructors and students), and “helps students become more proficient at programming.” We draw on Douce’s work to derive our categories for analyzing the practical value of an AAT. In 2010, Ihantola built on Ala-Mutka’s analysis with a review of AAT-related publications from 2006–2010, focusing on identifying variance in design aspects among the tools.66 In addition to providing substantive research and results, these articles present a wide-ranging collection of tools that we included among those we reviewed in our own study.

3. Methodology

To answer our four supporting research questions, we needed to find reliable reports on the use and experiences of AATs in courses. We identified hallmark AAT overviews addressing the development, uses, flaws, and popularity of automated assessment tools.64, 65, 66 We then identified a wider range of publications either citing or cited in these overviews. We reviewed over one hundred papers, initially tagging papers relevant to one or more of our research questions—improvement in student learning, student perceptions of AATs, faculty perceptions of AATs, accuracy or correctness of the tool—and excluding irrelevant papers. Regarding our
overall research question, then, we determined automated assessment tools “helpfulness” separately for each of our four categories.

3.1 Learning improvement

Research question 1 was: *Have AATs proven to be helpful in improving student learning?*

Relevant publications showed experiments carried out with a control group not using an AAT compared to an experimental group that was using an AAT. Ideally, all other course factors would remain the same between the two groups, and the composition of the two student groups would be similar. We also looked for smaller scale experiments that focused on a certain topic or unit within a course. Valid experiments could be performed either with the control and experimental groups simultaneously, or across different semesters as long as other factors were controlled as much as possible. We recognize learning improvement as shown by student improvement in a relevant assignment or exam, overall course grade, or results of third-party assessment tools. Increases in these categories would be evidence of the AAT being “helpful” in student learning.

3.2 Student impressions of and attitudes toward AATs

Research question 2 was: *Do students think that AATs have improved their performance?*

Students’ attitudes toward a course is a major determiner of how well they do in the course, and students’ acceptance of AATs is necessary for such tools’ successful widespread use. If students have an overall negative impression of AATs, they may eventually attempt to avoid courses that employ them. And unfounded student beliefs about negative effects associated with changes in the mechanics of a course often become self-fulfilling prophecies.

Publications we selected as relevant addressed student impressions based on their user experiences with AATs. In these papers, students were asked questions about the tools’ overall perceived helpfulness, ease of use, and fairness. Related questions addressed how well the students liked using the tool and whether the inclusion of an AAT in a future course would make the student more likely to take that course. Positive survey responses to these questions would indicate that AATs are “helpful” in contributing to positive student impressions and attitudes.

3.3 Instructor impressions and satisfaction with AATs

Research question 3 was: *After having used the tools, do instructors think that the tools have improved their teaching experiences?*

In the authors’ own discussions with other instructors, one of the first topics that comes up when discussing AATs is the expected time savings. Some instructors have expressed distrust in fully automated assessment, without a manual verification of the results. We have also heard concerns about anticipated additions of labor in setting up these tools and potential dangers from students trying to upload malicious code. We looked for published research in which instructors using
AATs have discussed such concerns based on their experiences. Instructor reports addressing these concerns with positive outcomes would indicate that AATs are “helpful.”

3.4 Measures of accuracy of the tools in assessing assignments

Research question 4 was: Is the assessment performed by AATs accurate enough to be helpful?

Do AATs grade assignments comparably to human graders? An AAT eliminates the possibility of bias toward certain students and inconsistency across multiple graders within a given course, but the accuracy of AATs should be quantitatively measurable. In order to assess this item, we reviewed studies in which the feedback given to automatically graded assignments was compared to the feedback given by human graders. Comparable grading would indicate that AATs are “helpful” in replacing human grading.

4. Analysis of results

4.1 Learning improvement

Overwhelmingly, our study revealed a positive impact on student learning with the introduction of an AAT into a course. Learning improvements were most often measured by researchers through either end-of-course grades or final exam scores because they are easy to collect. An improvement in the future may be the use of an independent concept area exam such as the Foundational CS1 (FCS1) Assessment Instrument.68

We discovered eight papers that reported on a positive increase in student learning that occurred when an AAT was introduced into the course.

In 2003, Edwards showed interesting results when switching from one AAT to another in a junior-level course on comparative languages.13 Originally, he was using an in-house AAT named Curator. Curator produced a set of random test cases for each submission that was reported back to the student with the appropriate expected output. Students were limited to five submissions. With increasing interest in test-driven development, Edwards switched to the AAT Web-CAT, which requires students to submit test cases when submitting their program to the tool. Using Web-CAT, students saw how well their programming submissions passed their own test cases and were given some feedback as to the coverage of their test cases. Because Web-CAT requires students to create their own test cases, Edwards did not put a limit on the number of submissions a student could make. Keeping other factors the same as much as possible, students using Web-CAT received, on average, a final course grade of a half a letter grade higher. But an even more interesting finding may be that those students started submitting programs (along with their test cases) much earlier with Web-CAT (an average of 4.2 days before the deadline vs. 2.2 days with Curator) perhaps due to the encouragement to submit earlier.

In 2003, Woit tested how different combinations of required and optional in-class assignments, online quizzes, and online midterms influenced students’ grades.69 Woit stated that weekly online quizzes provided better results than continuous required in-class assignments, but he
concluded that regular in-class assignments should still be required. Woit’s data showed how each of four studies affected grades and included student perceptions of the different methods. The data indicated that online midterms improved online final exam grades, while student perceptions indicated that online tests provided more accurate grades, motivated them to cheat less, and reduced the stress of the final. Data was collected from four different configurations of assignments that were given over a four-year period. The year in which students scored the highest (year 3 of 4) was the year in which they were given the most online practical quizzes through an AAT.

In 2005, Higgins described an experiment in which Ceilidh was replaced by CourseMarker at the University of Nottingham. CourseMarker improved grades after the parameters for assignments were tweaked in response to early results. From 1998–2001 and 1999–2002 respectively, the overall percentage of students passing first- and second-level programming rose. The authors do not provide specific numbers, but they clearly correlate student improvement to CourseMarker when they write, “The ratio of student passes to failures is very high, and has improved with the evolution of CourseMarker and the support provided by the system.”

In 2005, Kumar showed learning improvement with an automated tutor aimed at testing static and dynamic scoping concepts in a programming languages course. The author’s experiment consisted of a pre-test and post-test given within a short time period. In the experiment, three different groups of students were given either no practice between the two exams, a practice session with the tutor receiving minimal feedback, or a practice session with the tutor receiving extensive feedback. A positive correlation resulted between the increase of test scores and the amount of practice with detailed feedback from the automated tutor. When the tutor provided detailed feedback, student scores increased by a statistically significant (p<0.05) average of 50% the first semester the experiment was performed, and 58% the second semester. The weakness in the experiment is the small sample size: the first semester only contained seven students; the number of students for the second semester is not published.

Also in 2005, Malmi showed results from students using TRAKLA and TRAKLA2, in which end-of-course exam grades increased when instructors modified the ways in which students were allowed to use the automated tool. One major modification in 2004 was in allowing students to resubmit incorrect answers to the TRAKLA2 system by giving them a new problem that they had not seen before. This modification motivated the students to work longer on assignments and thus earn higher grades. However, Malmi also notes that resubmissions can both cause students to work carelessly during early submissions and also create a pattern of coding, resubmitting, and fixing in response to feedback without putting any thought into actions outside of what the tool suggests. Furthermore, Malmi observes that providing students with feedback on their answers, usually in the form of hints, did more to improve performance than providing example code and solutions. In the end, from 2003 to 2004, the percent of students receiving full points for their submissions increased from 10% to just over 30%. A very significant correlation (p<0.001) was shown between earning points on these automated exercises and earning points on exams.

In 2009, Towell introduced an automated assessment tool, Athene, to some of his CS1 courses. Athene takes student-submitted code and runs it against various test cases (dynamic testing) to give students feedback regarding the correct running of their program. In his study, 79 students
were in courses using Athene, and 46 students were in non-Athene courses. In the Athene-
courses, 71% of the students finished with a grade high enough to move on to the next course
(grade of A, B, or C), whereas in the non-Athene course, only 46% of the students scored high
enough to move on. One variable that was not held constant across the two different course
groups was the number of assignments given. A major goal of Towell’s was making it possible
to increase the number of assignments by decreasing the need for manual grading. With the use
of the AAT, students were given 75 assignments, while the non-AAT group were given only 15.

In 2011, Wang was able to show results from a controlled experiment that compared course
sections using an AAT to sections not using an AAT. Four course sections were taught in a
given semester with as little difference in learning material and delivery between the sections as
possible. In two of the sections, however, students emailed their code to a lab assistant, who
provided manual feedback, while in the other two sections, students submitted their code to the
AAT AutoLep. AutoLep performs both dynamic and semantic checking of student-submitted
source code and immediately provides feedback to the student. At the end of the course, all
students were given the same final exam, which was automatically graded. The two course
sections using the AAT finished with code that had a lower overall syntactic and structural defect
rate (5.6% vs. 8.7%), a lower functional error rate (10.1% vs. 13.5%), and a higher overall exam
score (79.3% vs. 73.2%). This experiment was one of the few we found that had a high level of
control across groups. Each group had a size of 60, had similar entrance scores, and took the
course during the same semester with the same teacher, textbook, and assignments.

In 2014, Falkner showed that increased granularity in grading with an AAT (with resubmission)
encouraged students to work longer to solve a programming assignment. In this context,
granularity refers to the number of unique feedback responses and/or grades that the student may
receive from the AAT. A significant increase occurred in the number of hours worked when
marking granularity was increased to 5 (from some value of 2–4). An additional increase in
scores occurred for students on a given assignment when the granularity was increased to a 15.
Falkner noted that the “students undertaking assignments with higher granularity schemes are
working over more of the post-deadline period, for longer, well after any on-time benefit of
additional granularity has stopped increasing, which we interpret as an overall increase in
persistence.”

Some published results, considered altogether, showed no increase in any learning measure. Two
published papers showed mixed results using the same tool, with the second paper explaining the
discrepancy as possibly due to not having a true control group in the first. We also include here
reports from a third experiment in which researchers expected to see some measure of student
learning, but did not.

Garcia-Mataesos included the introduction of Mooshak, an AAT commonly used in programming
contests, into an Algorithms and Data Structures course. The instructors were attempting to
decrease the dropout rate and, after introducing Mooshak, the dropout rate in the course did
decrease from 72.28% to 44.80%. However, the experiment was not conducted as a controlled
experiment focusing primarily on the introduction of an AAT, and the effect on the dropout rate
may be attributable to other simultaneously altered factors in the course. In 2014, Rubio-Sanchez
attempted to replicate this experiment with the introduction of Mooshak into a course. This
experiment was performed in a much more controlled manner, and the results showed no significant difference in the dropout rate between the Mooshak and non-Mooshak courses. The authors suggested that the decrease in dropout rate shown by the Garcia-Mataeos study was most likely due to simultaneous changes in course grading, specifically, implementing a new policy in which students who completed all assignments did not have to take the final exam.

In 2014, Denny reported on a tool that was created to enhance error messages that are generated by the compiler. Often compiler messages are difficult for students to understand. In Denny’s project, the researchers used the CodeWrite tool for Java programmers, intercepting the compiler error messages that the tool returned. The existing compiler error messages were replaced with much more descriptive error messages geared to the novice programmer. The results of the experiment showed that there was no statistically significant difference in the students’ behavior: students submitted as often as others had before to get past the same compile errors.

Given the above data, we refer back to research question 1: Have AATs proven to be helpful in improving student learning? We showed 11 relevant papers that had published data in addressing this question. Of these 11 papers, 8 showed positive increases in some measure of learning. Most of these measures were either final course grades or exam grades. Many of the experiments that were referenced did not include a large amount of detail as to the rigor of the experimental process, but some of the papers reported a very well carried out, tightly controlled experiment. There appears to be enough significant evidence in these 8 papers to conclude that student learning did increase. The 3 other papers showed no measurable increase or decrease in student learning after the introduction of an AAT. Given the results shown in these 11 papers, we conclude that AATs have proven helpful in improving student learning according to our criteria.

4.2 Student perceptions of automated assessment tools

Our study showed inconclusive results regarding student perceptions of AATs. Student feedback, almost exclusively through comments and surveys, has been frequently published. Some of these published results were very positive, but a significant number showed student dissatisfaction with the tools. We reviewed eleven papers that included positive student feedback regarding tools’ implementation or ease of use.

CAP is an automated static code analyzer for the Pascal language. It points out syntax and logical errors and attempts to give the student highly readable help right in the middle of the source code. A survey was given to 520 students at the end of the semester using CAP and the students were asked to rate CAP as a learning tool on a scale from 1 to 8 (8 being most favorable). The average student response was 6.85. Also, 11 out of the 12 instructors rated the student’s satisfaction with their grades and the grading criteria as improved.

In 2003, Edwards created a 20-question survey for students using Web-CAT, a tool that gives students the ability to submit and receive feedback on their own test cases. Edwards found that perceptions of using Web-CAT were generally positive. Perhaps the most telling response was to the question, “Even if it were not required, I would like to use Web-CAT to test my programs for class before turning them in.” This question received one of the most positive responses, scoring a 3.8 out of 5.
In 2004, Harris gave a survey to a population of 28 beginning programming students on the value of Submit in the course. To the question, “Submit only allows programs to be submitted if they have been verified as working correctly on all the test cases. How much did you like knowing that your program, which had been submitted for evaluation, had been verified as working correctly?” 89% of the students replied with a 3 or 4 response (on a 4-point scale, where 1 was a negative response and 4 was a positive response). To the question, “How useful have you found the diagnostics provided by Submit in helping you determine why your program was functioning incorrectly?” 72% of the students replied with a 3 or 4 response. And to the question, “How useful has it been to have had graded programs returned within a week of being submitted?” 88% of the students responded with a 3 or 4. Some responses to the open-ended question, “What one thing do you like about Submit?” included, “It assures my program is working and guarantees me at least an 80 on the programming assignment”; “The fact that it allows you to submit at anytime since it was activated”; and “I like the fact that it does test for every single case that could possibly arise. It covers some cases that I probably would not think of testing.”

Also in 2004, Ala-Mutka published on the use of Style++, an automated tool for assessing the style of a C++ program. In this context, poor style covers such areas as using global variables, performing floating point equality comparisons, and not including a reasonable amount of commenting. Over 1500 students had used the AAT, and 90% of those students agreed with the statement that “Style++ is a valuable aid.” One of the more interesting findings was that students seemed to rate the tool higher the more that they used it. On a 7-point Likert scale (higher being better), students who had only used the tool for one course rated the level of usefulness (for coursework and learning) at an average of 2.5, whereas students who had used the tool for 5 courses rated the tool at an average of slightly over 5. Students appreciated the fact that they could use the tool at any time and that they could get feedback from the tool before they turned in their assignments for grading.

In 2005, Higgins distributed a survey programming students who tested the tool CourseMarker after a transition away from Ceilidh. The surveys showed that over 75% of students appreciated features only AATs could provide such as multiple attempts. Specifically, most students felt that multiple available submissions encouraged them to work for a higher grade.

In 2007, Nordquist wrote on the tool Autograder, used for a beginning programming section and three secondary sections. Twenty-five out of 47 students volunteered to answer an anonymous online survey. This survey used a 1 to 5 Likert scale, and the responses reflected an overall positive response to the automatic grading tool. When asked, “Other things being equal, given the choice between taking a programming course with or without the autograder, I would choose the course that used the autograder,” the average response was 4.54. Eight out of the 20 responses to a freeform comment question referred specifically to appreciating the immediate feedback aspect of the tool.

In 2009, when Garcia-Mateos introduced Mooshak, he presented students with a survey designed as a series of questions prompting for agreement or disagreement. 77% of the students indicated that “they learn better with the new methodology than with the old one,” while 91%
said that “if they could choose, they would follow the continuous evaluation methodology again.”

In 2012, Watson discusses the tool BlueFix, which alongside other changes applied his principle of adapting the compiler messages to the level of the students. The students’ submissions improved in precision by 19.52% compared to submissions to the tool it replaced. All students viewed the enhanced error messages and fix suggestion capabilities of BlueFix as a useful aid to help with error resolution. Out of the 39 students who participated in the study, 56% and 59% believed it was difficult for a regular compiler to identify which programming errors had been made and isolate the location of the error respectively. Additionally, 81% of the students perceived value in including a social aspect that would allow them to discuss particular fixes with peers and instructors.

Also in 2012, Brown surveyed students using the JUG automated assessment tool on their perception of the tool’s impact. Given the question “Did the auto-graded tests match your expectations of the requirements?” the majority of students selected the middle answer, “Sometimes.” But the question “did the reports from the auto-grader clarify how your code should behave?” elicited a much more positive response, with the majority of students answering “Often.”

In 2013, Helminen presented a collection of student perceptions on the use of a web-based Python AAT. According to Helminen, “A great majority of the students, 79%, mostly or strongly agreed with the statement that the web-based programming environment should be used in the course assignments in the future (mostly 35%, strongly 44%).” A similar percentage of students liked the lack of setup of tools and environment. In the two free-response questions, 32 students gave general positive feedback about the tool, and 25 students specifically commented that the user interface was clear and easy for them to use.

Also in 2013, Holton reported on the most significant results of a student survey about use of Trillium, an AAT, on one problem in one of two computer science courses. Before and after using this tool, students were asked to agree or disagree with the statement, “A submission system that allowed me to test different solutions and provided rapid feedback on my homework performance would be valuable to me.” Over 80% of students agreed in both classes before and after, but, interestingly, 5% fewer students agreed in both classes after finishing their assignment than before they had started. Further, while over 85% of students saw some usefulness in systematic testing before their assignment, all students in one class saw its potential afterward whereas only 75% of students in the other class found it helpful after completing their assignment. Finally, 67% of the first class felt the feedback Trillium offered was helpful, whereas only 42% of the second class agreed. These numbers seem to correlate to the percentage of students that found the tool easy to use: 78% in the first class and 42% in the second.

Some papers reported student reviews that were either reluctant to test a new AAT from the aspect of the students’ grades, as is the case in the first two papers below, or indicating improvements that need to be made on the grading precision and feedback from the tool, which is true of the four papers that follow.
A 2000 paper reports that the adoption of the PGSE meant moving to a semi-automatic grading tool, and the students’ initial reaction was overwhelmingly defensive in response to the grades. They did not think it was fair to be penalized for “minor offenses” in their program that had previously been overlooked by the instructor. Jones makes the point that the instructor is just as responsible for the grading system as before in handwritten analysis, by having to carefully design a testable specification. He concludes that both the instructor and the students must make a commitment to the automated grader in order for the intended benefits of the AAT to be realized.

Also in 2000, Jackson related that a tool designed to work in tandem with a tutor or instructor met with success in decreasing the workload for the individual along with reducing the time taken to grade. With the tool, assignments hardly ever needed to be graded by hand to any degree, submissions could be analyzed more thoroughly, and grades were reported in a timelier manner. Jackson also noted the consistency in the grading of this system, since the tool minimized subjectivity. Interestingly, surveys reported that students felt hesitant to accept a tool that did not rely on a human to some degree.

In 2004, Harris included two open-ended questions in the survey mentioned in the previous section. Based on responses to these questions, he concluded, “Students did not like the precision that is required of them in the programming environment and the fact that Submit only highlights the first discrepancy between expected and actual output.” Students proposed that the tool should identify multiple errors at once to speed up the process of error correction. Other student complaints concerned the imprecise error messages. For example, one student wrote, “I don’t like the fact that our program has to look 100% like the program in Submit. It takes away from creativity and makes it feel like we have no creative powers within our own program.”

In the BOSS system, students were allowed to submit assignments multiple times without being penalized. However, complaints occurred about the system being too finicky regarding non-content related issues. Specifically, the complaints referred to extra whitespace in output resulting in low grades despite having the correct implementation. Also, students objected that BOSS’s grading scale was not fair to students whose programs were close to the solution, with only one or two errors in place, who still received a low grade.

In 2008, Suleman surveyed students using the Automatic Marker who responded with varied opinions on the efficacy of the tool. Just over 30% of the students did not find the grades given by the system fair, but Suleman attributed this to students likely not understanding the question (i.e., not understanding that “fair” meant that every student received grades assigned from the same scale). That portion of students may have believed the grades were too strict or arbitrary, disliking the scale itself rather than believing that students were marked differently. The tutors and a minority of students expressed dissatisfaction with the feedback provided. The author noted that the tool’s feedback was rather concise and did not display the test values that generated the error.

In 2014, Rubio-Sánchez issued a Mooshak questionnaire in which it became evident that students did not appreciate using the tool in general, despite regarding it as interesting or thinking its use was a good idea. The mode for item m12 (“Mooshak’s feedback is adequate”)
corresponds to “‘strongly disagree.” In the open-ended question, the most common critique was
related to Mooshak’s feedback, which students considered to be poor. Thus, after working with
Mooshak throughout the course, it seemed that students saw room for improvement in the tool
with regard to its error correction. This result is not surprising, given that Mooshak was created
to be a programming contest tool, which would not offer advice on incorrect submissions.

Given the above data, we refer back to research question 2: Do students think that AATs have
improved their performance? We examined 16 papers with published results of students’
impressions, attitudes, and experiences using AATs. Of these 16 papers, 10 reported primarily
positive student perceptions, while 5 mostly highlighted student concerns about AATs. One
paper reported feedback that was positive in some areas, but negative in others. The papers
reporting positive results showed that students thought the tools were helpful in giving them
quick and meaningful feedback, appreciated that the tools were constantly available for use, and
that they were easy to use. Multiple studies showed that students would like to use the tools in
future courses. Of the papers reporting negative feelings, the primary complaints were that the
AAT was too “picky”; i.e., that answers had to be too precise. Some students were also worried
that there was not a human looking over their results; these students expressed concern that the
AAT would be unable to evaluate minor errors in ways that humans would easily recognize.
Given the mixed results in these 16 papers, there is insufficient evidence to show that students
found that AATs have helped them.

4.3 Instructor perceptions of automated assessment tools

Overall, instructors appreciate AATs for the benefits they provide, such as the time savings.
Most instructors report they must invest time in larger quantities before a class first uses an AAT
compared to subsequent semesters, but the general consensus is that these tools are effective
time-savers and are proficient at the tasks they are designed to perform. Eight papers described
instructor perceptions of various increases in everyday productivity.

In 1995, Schorsch reported that 6 of 12 teachers who taught a class that used CAP to grade
assignments stated that the tool saved them around ten hours of grading per section of roughly
twenty students, in comparison to the previous semester’s workload with approximately the same
amount of students.8

In 2000, an unnamed tool studied by Jackson required human assistance in grading
documentation, tested ambiguous cases on rare occasions, and produced the final grade.27 This
tool could not assess documents meant to support the use of a submitted assignment; a tutor
could look over submissions to augment the system’s analysis in generating the grade if such a
need arose. Instructors reported that this tool reduced their workload for the course, particularly
because it reduced time taken to grade assignments and post scores. The system also allowed for
instructors to analyze submissions more fully, since they no longer had to focus on grading.
Additionally, the tool minimized subjectivity in the grading process, resulting in more consistent
grading.

Also in 2000, Jones included instructor impressions of the effectiveness of an AAT called
PGSE.16 He noted that teachers remained responsible for students’ grades, just as they were
without this tool, because the instructors designed the criteria for the scoring keys of assignments. This fact demanded that they remain diligent throughout assignment creation to ensure the quality of the automated grades. While this process demanded time initially, the tool allowed for the reuse of programs over time. Also, it encouraged instructors to manage their workloads when assignments needed to be created or tweaked, as the latter require significantly less time than the former.

In 2003, Venables stated that the feedback provided by Submit, the AAT she discussed, provided answers to many of the questions students would need to ask while working on an assignment. This AAT ability freed up class time that otherwise would have been needed for answering students’ questions. The tool also seemed to encourage students to research their problems or curiosities that resulted from assignments where the feedback did not meet their needs.

In 2005, Joy remarked in his publication of the results of BOSS that AATs should not require any more work than manual grading, in spite of the ability of these tools to allow for multiple attempts on a single assignment. About half of the students using BOSS took advantage of the ability to try again with a corrected submission, but the manager of the course did not see an increase in workload.

Also in 2005, Malmi, in discussing TRAKLA and TRAKLA2, pointed out that the time required to create assignments for an AAT, while initially intimidating to some instructors, was never repeated after the first use of the tool, since assignments only had to be tweaked for later semesters—a process which took comparatively minimal time. Additionally, students did not ask instructors and tutors as many questions, as the tools provided feedback that resolved much of the students’ confusion.

In 2012, Queirós briefly asserted that automated grading surpasses manual grading in efficiency, accuracy, and objectivity. AATs remove biases and other factors from the grading process, and submissions are marked in a fraction of the time humans would take to grade.

In 2013, Vihavainen reported that with his AAT, Test My Code, instructors were able to allocate their time toward tasks more productive than grading. In times they would have normally been grading, instructors were available to help students and complete various work-related responsibilities. Additionally, since teaching assistants no longer had to assist with grading, TAs could gain more experience tutoring students who needed assistance.

Two papers discussed specific improvements in the course brought about by the use of an AAT. In 1995, Schorsch demonstrated that 11 out of 12 instructors who taught a class before and after the use of CAP observed students became more satisfied overall with their grades after the introduction of the tool to the course. Also, teachers received fewer complaints about the grading criteria once the class started using CAP.

In 2009, Towell reported that Athene allowed instructors to assign approximately 70 homework programs, compared to approximately 12 from previous years.

Two papers provided insightful details of the usefulness of AATs based on their features.
In 2005, Joy noted the ability of AATs to provide students with quality test cases for their programs. Additionally, these tools could also be used to release as many test cases to students as the instructor desired, providing teachers with a simple way to control the amount of help they provided their students.

In 2008, Suleman reported that tutors were dissatisfied with the feedback provided by Automatic Marker. In elaborating, the tutors said the feedback was too concise and should have provided the test cases that generated the errors reported to students. Moreover, tutors expressed a concern that plagiarism of fellow students’ programs was more likely after the use of Automatic Marker in the course. They attributed the plagiarism to the likelihood of manual grading to catch plagiarism compared to the tool’s functional inability to flag similar programs.

In 2009, Murphy reported that Retina could track various statistics related to student submissions, such as time spent on an assignment versus grade, most common errors per assignment, number of errors per time of day, and other similar measurements. Instructors felt the statistics were useful in determining what concepts or types of programs each student would likely struggle with, in addition to the types of questions students would most likely ask.

Given the above data, we refer back to research question 3: After having used the tools, do instructors think that the tools have improved their teaching experiences? We examined 13 papers pertaining to instructors’ experiences with AATs. Instructors reported that these tools demonstrate tremendous time savings, lowering the burden of grading students’ programs so instructors can focus more on providing aid to students. Automated graders also calculate grades without the subjectivity and likelihood of human error present in human graders. Additionally, these tools decrease the amount of repetitive questions and complaints about due dates and grading criteria. AATs allow instructors to assign more programs and post grades more quickly. Such systems must be developed carefully, though, as feedback must adequately address possible errors. Often, instructors underestimated this burden when adopting an AAT, but once the tool was in place, instructors thought the effort required for initial set-up was worth the benefits. Instructors also expressed concern about an increase in plagiarism when using AATs and reported a need for plagiarism detection in these tools. Given the wealth of positive feedback, and the opinions expressed that the benefits of these tools outweigh the concerns, we determine that these tools have proven to be helpful to instructors.

4.4 Correctness of automated assessment tools

AATs have proven to produce useful results in aiding the assessment process. In this section, we report on the capabilities of AATs to produce accurate results. We reviewed six papers that discussed the ability of AATs to grade programs properly, identify algorithms, and check for program similarity.

In 2005, Higgins reported that CourseMarker successfully graded student submissions of one section of a course to at least the same level of accuracy as a teaching assistant responsible for the marking in another section of the same course. Higgins attributed the ability of the AAT to surpass the consistency of student graders to the tool’s lack of subjectivity.
In 2007, Wang was able to show the capabilities and shortcomings of AATs based on semantic similarity-based grading. The tool in this paper correlated with precision at or above 90% for at least 75% of submissions for 9 out of 10 tasks it graded. However, as this tool’s grading approach relies on model solutions, more complex problems require more example solutions in order to adequately assess a submission. When assignments reach a higher level of complexity, this method becomes too time-consuming to justify continued use.

In 2012, Jurado reported on grading consistency rates between AATs and instructors. When an assignment was graded both by an AAT and a human grader on a scale from 0 to 10, 43.42% of assignments differed in score between graders by less than .1 (on a 1–10 scale). And 92.04% of the assignment grades differed by less than 1 (on a 1–10 point scale). When the human grader knew beforehand the grade assigned by the tool, the average assignment grade between the tool and the human differed less than .1 (on a 1–10 point scale) for 47.25% of the assignments and by less than 1 for 76.78% of the assignments. Jurado speculates that these differences would lessen if AATs could better assess programming style, as the tools would then be able to grade based on human patterns and preferences without being affected by subjectivity.

In 2012, Taherkhani demonstrated that for about 75% of submissions, AARI was able to successfully identify the algorithms students used in a program that required them to sort integers in ascending order. This tool struggled most in identifying merge sort and algorithms that it was not designed to recognize. It identified insertion sort, selection sort, bubble sort, and quicksort with near-perfect accuracy.

In 2013, Singh showed that the AAT he discussed accurately found 64% of incorrect student submissions. He noted the need to refine the tool for greater reliability. However, he believed that with the corrections he suggested, this tool could be used to grade homework assignments for online courses and MOOCs.

In 2014, Gaudencio reported that instructors who manually graded assignments tended to agree more with the results of an AAT than with results other instructors provided. An instructor agreed with the grade given by the tool in 75–97% of cases, whereas an instructor agreed with the grade assigned by another instructor in 62–95% of cases. This agreement shows, he asserts, that AATs can be used effectively to grade code.

Given the above data, we refer back to research question 4: Is the assessment performed by AATs accurate enough to be helpful? This question proved to have the least amount of published data available, as we found only 6 relevant papers from which to draw a conclusion, but these papers documented worthwhile experiments that compared the grading of assignments by AATs to the manual grading of assignments. In some cases, the human graders knew they were reviewing grades already assigned by an AAT, but in other cases, they did not. In both cases, the human graders assigned grades similar to those assigned by the AATs. Instructors also spoke positively on the issue of reduced grading bias with AATs and increased grading consistency. The data presented shows that AATs can indeed grade at a level similar to human graders. While more study needs to be done in this area and more examples published for the community to see, enough positive data exists to show AATs can grade accurately enough to be helpful to instructors.
5. Conclusion

Are automated assessment tools helpful in programming courses? Given the literature that is currently available—specifically the research cited in this paper—we have answered our four research questions as follows:

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have AATs proven to be helpful in improving student learning?</td>
<td>Yes</td>
</tr>
<tr>
<td>Do students think that AATs have improved their performance?</td>
<td>Inconclusive</td>
</tr>
<tr>
<td>After having used the tools, do instructors think that the tools have improved their teaching experiences?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is the assessment performed by AATs accurate enough to be helpful?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The only question without a positive answer is that regarding student perceptions of AATs. To confirm that students’ negative perceptions are related to AATs specifically, we recommend performing more rigorous comparisons of student perceptions in courses with and without AATs, to see if the negative perceptions (“too picky”) are correlated to the course or to the tools. Instructors may feel that highly detailed (“picky”) scoring is justifiable based on the iterative development afforded by AATs and may feel justified in requiring greater precision, especially when AATs have generous resubmission policies. Depending on results of such research, continued work might then focus on improving AATs specifically to decrease negative student perceptions.

Considering together the large number of papers published about automated assessment tools, one noticeable point is that only a small percentage of those papers include formal results about the effects of tool use. Future work in this area should cover more rigorous experiments in effects on student learning, ideally with a shared deterministic metric for measuring learning (i.e., a standardized concepts exam). More experiments should also test comparable scoring. As our community of researchers creates, modifies, and uses these tools, we need to put more effort into quantitatively measuring the impact of AATs.

Given that AAT tools have proven to be helpful in programming courses, researchers should work together to lower the barrier of entry for new instructors to begin using these tools. We should work together across universities and across countries to promote and share best practices. There is much duplicated effort in writing assignments, test cases, and testing code for tool-specific problems. Through more intentional collaboration, we can leverage our work in more efficient ways to produce better tools to help current and future students.

References


[34] CodeLab® is a registered trademark of TuringsCraft at http://www.turingscraft.com/
[50] Draylson Micael de Souza, Jose Carlos Maldonado, and Ellen Francine Barbosa. 2011. ProgTest: An environment for the submission and evaluation of programming assignments based on testing activities. In


[72] Nickolas Falkner, Rebeca Vivian, David Piper and Katrina Falkner. Increasing the Effectiveness of Automated Assessment By Increasing Marking Granularity and Feedback Units. SIGCSE '14.


