Chapter 12
Objective Tests in Automated Grading of Computer Science Courses: An Overview

Marin Lujak, Marija Slavkovik, Alexis Lebis, Mathieu Vermeulen, and Arnaud Doniec

Abstract In this chapter, we analyze and mutually compare the most used objective tests in computer science courses in Learning Management Systems (e.g., Moodle) and MOOCs. We outline their advantages, technical limitations, and ethical challenges. We also consider test feedback mechanisms that facilitate continuous learning for students as well as the identification and recognition of possible evaluation mistakes of the system of AI-supported methods for automating objective tests in programming. These tests come with a range of technical challenges to promote students' empowerment and support their autonomy in the learning process. There are also ethical challenges that arise when a human evaluator is replaced by software. We discuss and analyze these tests and focus on identifying and mitigating the context-specific ethical challenges. A clearly defined form of evaluation also checks and promotes students' integrity and genuineness in summative evaluations. Contrarily, in formative evaluations, automated objective tests should help to motivate students to commit to the learning process. We emphasize the essential characteristics for these tests to correctly provide these two types of evaluations in courses.

M. Lujak (✉)
CETINIA, University Rey Juan Carlos, Madrid, Spain
e-mail: marin.lujak@urjc.es

M. Slavkovik
Department of Information Science and Media Studies, University of Bergen, Bergen, Norway
e-mail: marija.slavkovik@uib.no

A. Lebis · M. Vermeulen · A. Doniec
IMT Nord Europe, CERI Numerique, University of Lille, Lille, France
e-mail: alexis.lebis@imt-nord-europe.fr

M. Vermeulen
e-mail: mathieu.vermeulen@imt-nord-europe.fr

A. Doniec
e-mail: arnaud.doniec@imt-nord-europe.fr

M. Ivanović et al. (eds.), *Handbook on Intelligent Techniques in the Educational Process*,
Learning and Analytics in Intelligent Systems 29,
https://doi.org/10.1007/978-3-031-04662-9_12

12.1 Introduction

The demand for professionals with programming skills and most recently AI skills is constantly growing. We observe an ever-increasing number of students in both face-to-face and online engineering and programming courses. Yet, the number of lecturers and teaching assistants does not keep pace with the student demand. As a result, the number of students per course increases, leading to massification issues most evident in the Massive Open Online Courses (MOOCs). One of the issues is related to student dropout and disengagement, mostly due to the lack of motivation (see, e.g., [80]).

Interlaced with this problem is the issue related to the effective and continuous assessment of the students' progress in time. In order to determine students' competency in achieving curriculum expectations, summative tests assess what has already been learned and is often done at the end of the unit, period, or course. Ultimately, the goal is to assure the assessment is meaningful to support well-designed lesson planning. The assessment should be both consistent and valid as well as fair to all students. By fair, we mean impartial and just without favoritism or any kind of discrimination.

Technology can assist us in providing authentic cross-curricular assessment through the use of online education platforms as an aid to the teaching in the classroom as well as a support of the courses held completely online through, e.g., Learning Management Systems (e.g., Moodle[1]) or MOOCS. In such platforms, objective tests are the most common way of student evaluation. They are automatically evaluated providing interpretive reports. The strengths of automated tests, among others, include objectivity, consistency, speed, and 24 h availability [4]. Other ICT technologies in assessment that are out of the scope of this chapter include AI-based robotic process automation, machine learning, pattern matching, and natural language processing; all of these giving new perspectives to the Technologies Enhanced Learning (TEL), especially for exam and progress evaluation (see, e.g., [39]).

In this chapter, we analyze and mutually compare the most used objective tests in computer science (CS) courses taught online. We outline their advantages, technical limitations, and ethical challenges. We also consider test feedback mechanisms that facilitate continuous learning for students. These tests and their feedback mechanisms come with a range of technical challenges to promote students' empowerment and support their autonomy in the learning process.

There are also ethical challenges that arise when a human evaluator is replaced by software (fairness, transparency, bias, explainability, etc.). Numerous ethical guidelines and some laws (e.g., GDPR in the EU) are being proposed to ensure that the use of personal data and automation of human tasks are done ethically. Of particular concern is the possibility for an automated process to introduce or reinforce bias in society, or operate unjustly, cutting stakeholders and users from the ability to correct introduced mistakes. Although some general principles can be outlined for all TEL

[1] Moodle; http://www.moodle.org. Accessed 1 July 2021.

solutions, there remains the need to discuss and analyze them in a specific context and focus on identifying and mitigating the context-specific ethical challenges.

Nonetheless, objective tests serve more roles than just establishing whether the student has given a correct answer to a problem. A clearly defined form of evaluation also checks and promotes the integrity and genuineness of a student (e.g., prevent a student gaming the system) in a summative evaluations context. On the other hand, in a formative evaluations context, objective tests should help motivate students to commit to the learning process. We emphasize the essential characteristics for these tests to correctly provide these two types of evaluations in computer science courses.

Our methodological approach to this overview is drawn from three main concerns from the point of view of the students and the teachers using objective tests in automated grading of computer science courses: their advantages, limitations, and ethical challenges. As such, this chapter is not intended as a review of the current state of research in objective tests but is rather an overview capturing key aspects of objective tests in automated grading of computer science courses. It is intended both as introductory material for the researchers in the field as well as for the teachers using objective tests in automated grading of their computer science courses. The methodology encompasses objective test development and design activities, a description of challenges to sustain their efficient and effective use, and applicability criteria outlining the kind of learning outcomes and tasks that can benefit from them. This chapter is organized as follows. In Sect. 12.2, we give an overview of learning assessment and online objective tests in automated grading of computer science courses. We discuss the differences between assessment and testing, and subjective versus objective assessment. We give five principles of testing and assessment that will be used as key performance indicators in the evaluation of objective tests. In Sect. 12.3, we categorize objective tests, while in Sect. 12.4, we discuss bias, ethical issues and explainability in objective tests in automated grading of computer science courses. Finally, in Sect. 12.6, we consider open challenges and discuss the requirements for future proactive automated grading systems.

12.2 Background

Student learning outcomes are the knowledge, skills, attitudes, and values that students obtain in a course. These outcomes identify the obtained student's abilities and aptitude related with the learned material at the end of the course as well as the value of the learning process. Based on the hierarchy of cognitive-learning levels of Bloom's taxonomy [6, 16], cognitive domain can be divided in: (i) knowledge of specific facts and conventions that is the lowest-level of learning shown through remembering of previously learned material, (ii) comprehension or understanding, (iii) application or the ability to use the learned material in a new and concrete situation, (iv) analysis or the ability to break down the learned material into its components to analyze their relationships and to recognize underlying organizational principles, (v) synthesis or the ability to resolve contradictions and to create a new whole by

mutually assembling the parts, and (vi) evaluation or the ability to judge the value of compiled material for a given purpose.

The necessary cognitive skill required in higher-education computer science courses is to understand basic programming concepts, principles, and paradigms, i.e., the main constructive elements for writing program code. However, understanding the concepts and principles does not assure the capacity to write computer programs, which requires extensive practical experience and skills. To acquire them, a student should develop a higher-level knowledge by applying the guiding concepts, principles, and paradigms on simple programming tasks, analyze the code for debugging tasks, and synthesize the code in advanced programming assignments.

12.2.1 Differences Between Assessment and Testing

Testing serves to determine final grades or to make promotion decisions, to identify the areas that need improvement or stronger emphasis or where the curriculum is weak, to motivate students to study and to communicate to students what material is important (see, e.g., [58].

Conventional classroom tests are performed in a given limited time in a written or oral form to measure the learning process at a specific point in time. Assessment is a broader concept of testing where a teacher uses a range of continuous-learning procedures to evaluate the learning process and includes both formal and informal observable measures for judgement (see, e.g., [15, 61]. It is a continuous process in which measurable and clear student learning outcomes determine how well student learning matches expectations (see, e.g., [19, 89]).

We can distinguish formative (used during the teaching process), summative (measuring long term academic goals, mostly used for grading, e.g., midterms and final unit tests and projects), and diagnostic assessments (used to get prior knowledge on students and to plan future instruction, e.g., a test before a course).

Formal assessments are strict and specific testing procedures and rules, e.g., a standardized test such as the SAT), while informal assessments lack supporting data and use normal classroom testing procedures. Performance-based assessments assess student's ability to complete work in an academic task, e.g., the design of an application for a computer programming class.

From the point of view of administering the assessment, it can be an individual or group assessment. Norm-referenced assessments are based on a comparison of students against similar demographics.

12.2.2 Objective Versus Subjective Tests

Generally, we can divide test questions into objective and subjective ones. In an objective question, a student should select the correct response from several alternatives or provide a short answer or complete a statement. Subjective questions, on the other hand, allow for an extended original answer to a question in terms of an essay, a problem solving or performance answer. Choosing between objective or subjective tests depends on the learning objective that we want to measure.

Automated assessment requires questions that can be marked objectively. There is no objective assessment in the absolute meaning of that word. Each assessment performed by a human is made with inherent bias built into decisions about relevant subject matter and content. If a teacher is not cross-culturally sensitive, further bias related with socioeconomic status, age, disability, cultural background, ethnicity, religion, gender, and sexual orientation may occur (see, e.g., [53, 72]).

However, in addition to these biases, subjective assessment is further limited by individual evaluators that can rate the same exam differently over time. The lack of anonymity as well as handwriting and grammar can affect the grading process (see, e.g., [71]).

The presence of marking bias requires either the use of multiple assessors for each student, or standardization of marks between assessors, to minimize the problem. This problem is related to intra- and inter-rater reliability.

For a teacher, subjective questions are generally easier and faster to construct than most objective questions. Students with good writing skills will generally perform better in subjective tests. Moreover, due to the extent of time required to respond to a subjective question, the number of the questions in a subjective test is generally much lower than the same in an objective test. In the case of lack of penalties, objective tests can be correctly answered through blind guessing, while subjective tests can be partially responded through well-written bluffing.

For some learning purposes, objective or subjective tests may prove more efficient and appropriate but both can measure similar content and learning objectives. Students respond almost identically to both tests covering the same content (see, e.g., [42]).

12.2.3 Five Principles of Testing for Assessment

The five commonly recognized principles of testing and assessment are: validity, reliability, authenticity, practicality, and washback.

Validity

There are multiple definitions of validity in the literature that seem to be evolving (see, e.g., [64]). One of them is that validity considers the extent to which inferences made from the scores on a test are appropriate, meaningful, and useful for the purpose of the test [55]. Airasian [2] defines validity as the degree to which assessment

information permits correct interpretations of the desired kind. In general, validity concerns accurate measuring of what we aim to measure through test results; it is a measure of fitness for the decisions that will be based on those scores.

Reliability

Reliability is related to the degree of consistency of test scores or the degree to which they are free from various types of chance effects and errors of measurement (see, e.g., [24]). Reliability considers the likelihood that a student's score would change if that student repeated the same test or took another version of the test.

Reliability can be classified into *student reliability* (factors that affect a student's state of mind in the testing process resulting in an inaccurate test score), *intra-rater reliability*, i.e., how consistently the teacher scores individual tests in the class due to, e.g., fatigue, imprecise marking criteria, or student bias, *inter-rater reliability*, i.e., how criteria change from one teacher to another evaluating the same class tests, and *test reliability*, or how much, if a test is to be taken more than once, two or more equivalent forms of the same test mimic one another while being sufficiently different to avoid for the student to look up the answer.

Test reliability can be improved by clear task instructions. If the instructions are clear, the marks will be more consistent because the task makers know what to expect. Clear assessment criteria and scales are another way of increasing reliability because if the raters know what they need to focus on in marking, they will have high levels of agreement. The reliability is also increased by training teachers/examiners to deliver tests and score tests in a consistent way.

Clear assessment criteria and scales are supported by an analytic scale that breaks down the holistic scale based on certain criteria, and it facilitates reliability as the teachers can follow the scale in marking a test taker.

Authenticity

Authenticity relates to how closely tested material matches authentic challenges and conditions of real-world applications that students will encounter outside the classroom, i.e., authentic tasks in authentic contexts closely related to work done in the real world that students may engage in once they leave the class. Authentic tests involve engaging problems or questions of importance, error-free representation of the real-world context or field of study, and challenging tasks, among others (see, e.g., [97]).

As an example, problem-based learning (PBL) develops higher-order thinking skills by providing students with authentic and complex tasks that take place in real-world settings. However, multiple choice questions are not a particularly authentic testing method in checking the programming knowledge nor are they appropriate as predictors of real-world training criteria.

Practicality

Practicality relates to test logistics such as development time, scoring time, budget, resources, feedback delivery and administration issues including the means of test evaluation. A test is practical if it respects budgetary limits, it can be performed

entirely within the given test writing time, requiring not more than available human and material resources for both design, test monitoring, and scoring (see, e.g., [20]).

Washback

Test impact or washback relates to the direct effect of the test on learning and is the feedback given to students following a test or assessment. The best way to improve the impact is to test those abilities whose development you would like to encourage and not necessarily what is easiest to test. The challenge here is transmitting the results in a positive way that does not overwhelm the student in the most adequate form.

Formative feedback is aimed at helping students to improve their skills. Keuning et al. [63] give a systematic literature review of automated feedback generation for programming exercises in 101 tools to find out what kind of feedback is provided, which techniques are used to generate the feedback, how adaptable the feedback is, and how these tools are evaluated. They argue that feedback mostly focuses on identifying mistakes and less on fixing problems and taking a next step while the tools are not easily adaptable by teachers to their own needs.

To conclude, an effective and efficient test should test what it sets out to test (validity) while producing scores that can be trusted (reliability) and supporting learning (positive impact). However, it should be practical to develop and deliver.

The gender and background of the learner as well as personality traits could affect the way they perform (see, e.g., [85]).

12.2.4 Online Educational Assessment

Both examiners and students usually prefer online testing because of the convenience of the online platforms.

The ever-increasing number of online courses and MOOCs requires the use of techniques and methods to assess online learners. Although the use of computers brings many benefits for teaching [14], the very context of online education makes online assessments problematic and difficult for teachers. Indeed, learners are increasingly more numerous (often several thousand in the case of MOOCs), and they are in a distance learning context, both distant from other learners and teachers. Therefore, it becomes difficult to assess some elements which were first rather easily observable, like learner's participation or effectiveness regarding a specific task (e.g. is it really the learner who performed the task?). Furthermore, the formalism induced by learning systems hinders the evaluation of learners' competencies. Common questions we face when setting up such online courses are for example how could several dozens of a learner's open text responses be evaluated or how could soft skills be assessed in online training as well?

These issues are emphasized by the fact that classical assessments have been carried over online courses. Notable among those is the evaluation by tests that reports the result of a reasoning process (answer to a problem, closed question, etc.).

They are poorly suited to the assessment of skills and give little to no information about the reasoning behind the given result. However, online grading is fast during the actual grading phase, but some of this benefit is offset by the additional overhead prior to online grading [23].

There is also online homework, which is a more complete approach than a simple test since it requires learners to produce an authentic document. However, due to the context of online courses, they are difficult and time-consuming to set up for mass training courses, and this raises important questions regarding learner's honesty.

Consequently, several new modalities have emerged these last years to tackle these issues, mostly shifting the focus of examination towards the students. It is the case of ePortoflio assessments for example, which propose an increased range of evidence such as video, text or even audio, to facilitate assessment and bring meaningful information for learners [99]. Nonetheless, widely used modalities are peer evaluations (e.g., [60, 92, 100]), where learners contribute to the grading of other learners, thus increasing their engagement and mobilizing their skills, particularly in MOOCs [74], and self-assessment were the learners grade their own assignments. Additionally, these two modalities bring strong formative feedback [93] and are already in use, for example, in Coursera and EdX.

Moreover, researchers in Learning Analytics (LA) and Educational Data Mining (EDM) explore the new possibilities offered by the online educational environment by gathering and analyzing learner data in order to answer assessment issues. Through this data, it is possible to gain knowledge regarding learners' knowledge and behavior, as well as bringing new feedback both to learners and teachers [47, 95] and even grade them based on their activities [50]. While EDM relies heavily on the machine to perform analyzes (e.g. clustering), LA is a more involving process per se, and attempts to include at least teachers in the analysis and thus, presumably, better encompasses pedagogical expectations [45]. Either way, EDM and LA are useful for proposing indicators [34] and/or dashboards [29] to teachers and/or learners, which are profitable to understand a learning situation. These approaches are particularly interesting because many learners' activities can be observed, particularly those that are spread out over time (e.g. mobilization of soft skills in group work) and they have a high prospect regarding the sharing and the re-usability of analyzes across online courses and online systems [67, 78].

Finally, it has to be noted that the context of the COVID health crisis has highlighted once again the limitations of conventional assessments modalities in dematerialized environments [79]. The most notable limitations teachers face are an important amount of plagiarism and cheating from their students, thus preventing teachers from assessing students' skills, and the difficulty to adapt their tests and evaluation for online teaching is mostly due to the lack of guidelines. Nonetheless, the use of automatic processes to provide automatic correction or feedback in online teaching is steadily becoming more important [9], and maybe a part of the solution.

12.3 Automated Objective Tests

An objective test is a test consisting of factual questions requiring usually short answers that are either correct or incorrect. These answers can be quickly and unambiguously scored by anyone with total reliability with an answer key, thus minimizing subjective judgments by both the person taking the test and the person scoring it.

Objective tests are most suitable for use at lower-order cognitive levels such as memory, basic comprehension and application of numerical procedures but it is also possible to design these tests to test higher-cognitive levels.

Objective assessment is a generic term referring to tests that can be marked with complete reliability as there are clear right and wrong answers. Objective assessments can be summative, where marks are used to calculate a final grade for the student, or formative, where student efforts are met with feedback on their performance that does not directly contribute to final grades.

Objective tests can be set to test a wide topic area at any required level of difficulty, where large numbers of questions may be answered in a relatively short time covering widely course content and focusing on specific knowledge and skills although sometimes superficially. They are especially useful with large student groups where the test can be reused.

For large classes, it represents an efficient way of testing a large number of students rapidly and within short time frames, particularly when computers are employed to assist marking. As with all forms of assessment it is necessary to align assessment with the desired learning outcomes for the course.

The strength of objective tests is fairness, impartiality of evaluation, and no scoring variation due to the evaluator's fatigue or lack of student anonymity. Also, the strength is fast reporting of scores. Impartiality assumes equal treatment of all students and implies that the evaluation is based on objective criteria, and not on the basis of bias, prejudice or any other method that would benefit one student over another for improper reasons.

However, there are three major weaknesses of objective testing: if a bank of questions is missing, initially, it is very difficult and lengthy to construct it; test questions require to be tested before they are used in order to check their validity; and there are also known issues with gender differences in student performance of objective tests since, on average, women have a higher aversion to risk-taking in attempting to answer a question when a student is not certain of the result (see, e.g., [5, 25].

12.3.1 Types of Studied Objective Test Questions

It is important to note that there is no type of test that is intrinsically good or bad; each one has its advantages and weaknesses that should be evaluated in considering the fitness for the purpose of the test. However, concentrating on computer science

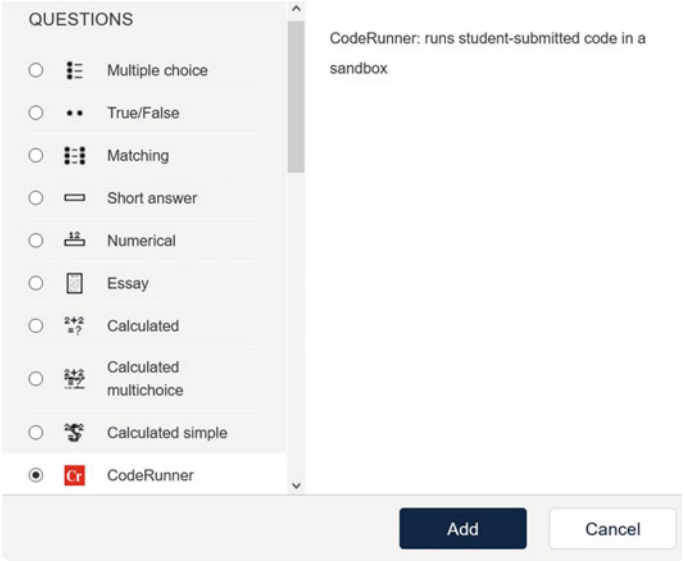

Fig. 12.1 Some question types in Moodle

courses, we study in this paper the following objective factual tests available in Moodle: multiple choice, true/false, matching, extended matching, and short response tests as well as automatic grading systems for programming (CodeRunner among others), Fig. 12.1.

We briefly present in the following each one of them and give their main characteristics. Their more detailed description can be found in, e.g., Gordon et al. [54].

12.3.1.1 Multiple Choice Questions (MCQ)

This most widely used type of objective tests allows the selection of a single or multiple responses from a predefined list. The test taker is usually given a short question (the stem) with usually four to five optional answers of which one or more are correct (the key) and others wrong (the distractors). A test taker must choose the right one(s) (see, e.g., [57]).

One of the advantages of multiple-choice tests is that questions with different levels of cognitive processing including higher-order thinking can be created. For example, we can measure reasoning, comprehension, application, analysis, and other complex thinking processes. Tasks are fairly easy to mark automatically online. The weakness of this test method is that there is a high chance of getting the correct answer through guessing [32]. It assumes that the world is fixed,there is a fixed number of answers and only certain ones are correct. The differences between the answers may be so small that a high proficiency of the context may be required, the context that is not usually strictly related to what is being tested.

Answering a multiple-choice question is not necessarily authentic since, in real life, the test taker will not choose a multiple-choice option. Additionally, the set of answers may be incomplete or misunderstood by the test maker based on the cultural background or the proficiency in the language the test is written in. Regarding bias, this kind of test may favor male students since they are statistically more likely to be risk takers (see, e.g., [13]). Therefore, this kind of test should be balanced with other test types. For an overview of multiple-choice tests (see, e.g., [7, 94]). McCoubrie [73] present the guidelines for the construction of multiple choice questions tests while [3] give a review of the literature treating the topic of improving the fairness of multiple-choice questions.

12.3.1.2 True/False Questions

A true/false test is a simple form of multiple-choice question with two choices: true and false. It can be divided into the tests where an examinee must select all options that are true and a single best answer.

Selecting all options that are true requires additional judgement and guessing in case the options are either not completely true or not completely false. This is why options must be absolutely true or false without doubt.

Even though the true–false tests are often considered as superficial tests that lack the pedagogical efficacy of more substantive tests, Brabec et al. [18] found that especially when carefully constructed, true–false tests can elicit beneficial retrieval processes that resemble those of other types of tests. Such benefits, however, might not be as consistent or powerful as those elicited by other tests [83].

12.3.1.3 Matching Questions (MQ)

This is a variation of a multiple-choice test where the answer to each of a number of sub-questions must be selected from a list of possibilities. Each item consists of two lists of statements, words, symbols, or numbers which have to be matched one with another. In general, the two lists contain different numbers of entries, those entries in the longer list that do not correspond to entries in the shorter list serving the function of distractors.

12.3.1.4 Extended Matching Questions (EMQ)

Extended matching tests are similar to multiple-choice tests. Given are at least two different scenarios and an answer option list where the options may be used once, more than once, or not at all.

This kind of test allows for an in-depth test of knowledge by providing scenarios similar in structure and content related to the question theme with one'best' answer

from the plethora of answer options given (see, e.g., [98]). There is a greater chance of answering incorrectly if an examinee cannot synthesize and apply their knowledge.

12.3.1.5 Short Response Questions (SRQ)

Short response or short answer tests require a direct answer made of one or a few words rather than simply chosen from a number of options provided. A short response test may consist of incomplete statements where the examinee must supply the missing pieces of information, e.g., a word. phrase or a number.

It is graded by comparing against various model answers, which may contain wildcard characters.[2] The problem with the use of the wild card character is that the number of answers considered correct is potentially infinite.

Short response tests can be used for both formative and summative assessment. Generally, they have high reliability, focusing on specific knowledge or skills and are relatively quick and easy to mark. Also, in the case of unique answer questions, the examinee has to provide an answer to a question in terms of a word or a number.

12.3.1.6 Automatic Grading Systems for Programming

Among the program features that can be automatically assessed, there are: (i) functionality (checking that the program functions according to the given requirements by running the program against several test data sets), (ii) efficiency, e.g., measuring running time of the program during the execution and comparing it to an existing model solution, and (iii) student's skills of designing test cases and testing thoroughly before submitting the program, (iv) programming style and design, just to mention a few. However, since all programming skills cannot be assessed automatically, manual inspection of some skills is still necessary (see, e.g., [4]). In this case, some tools can help to save time especially when teaching many students. In [28], for example, the authors propose to identify equivalences between algorithms submitted by students and then to cluster similar answers requiring similar feedbacks.

Over the past few years, many MOOCs in programming were created offering learners an integrated environment to do all the exercises directly in a browser. This type of MOOC has two main advantages: from the learner's point of view, it avoids tedious installation and configuration of non-user-friendly software (compilers, code editor, versioning system, etc.), which is often the first obstacle for beginners in programming; and from the teacher's point of view, it allows to implement an automatic grading framework.

Such an automatic grading framework must meet several objectives: allow the teacher to detect errors and deduce a grade, avoid cheating, allow the learner to get continuously quick and readable feedback, guide the student in solving algorithmic

[2] A Wildcard Character is a Kind of a Placeholder Represented by a Single Character, E.G., Asterisk (*), Interpreted as a Set of Literal Characters or an Empty String.

problems and writing code. Most experiments using automatic grading in programming MOOCs have used a well-known practice in software engineering: test-driven development (TDD) [10]. TDD consists of writing unit tests describing the functionality to implement before any attempt of coding. A unit test is a piece of code that checks the correct operation of a source code supposed to do something. Most programming languages provide frameworks for the implementation of unit tests, e.g., JUnit for Java, CUnit for C, and PyUnit for Python.

Transposed to teaching with MOOC, this principle requires the teacher to describe the expectations for each exercise in the form of unit tests. Depending on the programming language (and the associated unit test library), these unit tests allow us to evaluate the validity of the solution produced by the learner's code and in some cases the quality of the code.

Works presented in Canou et al. [22] show it is possible to create an automatic grader with the same assessment refinement as a real teacher. The authors succeed in evaluating the coding style of a learner checking for example the use of nested loops when necessary. They were also able to evaluate the algorithmic complexity of a code. Since using a running time limit cannot always be reliable (depending on the machine load), they use a tuned execution environment able to count the number of performed operations.

From a technical perspective, implementing automated grading is challenging. Two approaches are possible. The first one consists of running automated grading on a server; this means that the code written in the learners' browsers is sent to the server to be run over unit tests. This execution must be done in an isolated environment to prevent any malicious intent from a malicious user, e.g., memory exhaustion or extra CPU time consumption. The second one consists of executing automated grading directly in learners' browsers. This solution allows distributing the computation cost over all the learners and to avoid the possible attacks mentioned previously.

Romli et al.[82] give a review of approaches that were implemented in various studies regarding automatic programming assessment, test data generation and integration of both of them up till 2010.

Caiza and Del Alamo [21] reviews tools for automatic grading of programming assignments. The review includes the definition and description of key features e.g. supported languages, used technology, infrastructure, etc. emphasizing improvements on security, support of different languages, plagiarism detection, etc. emphasizing the lack of a grading model for assignments in the reviewed tools. However, there is no common grading being applied. Thus, further research is required to propose an automatic assessment that grades the student achievement based on learning taxonomy such as Bloom Cognitive Competency model [66].

In the following, we mention some state-of-the-art systems for automated testing of code. The aim is not to give an exhaustive list of the software but to give a short overview based on the main characteristics of the frequently used software.

Web-CAT[3] is a free, open-source automated grading system that grades students on how well they test their own code [43]. It is highly customizable and extensible, and

[3] https://web-cat.github.io/projects/Web-CAT/. Accessed 8 January 2021.

supports virtually any model of program grading, assessment, and feedback generation. It supports student-written tests, measurement of test coverage, and grading on test thoroughness. It also supports static analysis tools to assess documentation and coding style. In the static analysis, the student's code is compared with the model programs given by a teacher: the closer the student code is to the model programs, the higher is the mark. It also supports manual grading with direct on-line markup of assignments.

CodeRunner[4] [38, 69] available in Moodle is another free open-source question-type plug-in tool that allows submitting code as a solution to an assigned question and assesses computer programming skills by compiling and executing student-submitted code in a sandbox in different possible programming languages. CodeRunner has also been used in computer science and engineering for automatic test grading when multiple correct answers are possible.

The output is compared to the solution provided by the teacher. A student receives his/her test-case results and feedback immediately and is given the opportunity to modify the solution and resubmit, typically with a small penalty. This encourages students to learn through an iterative process until they achieve the correct solution. Such a system lifts the workload from evaluators and provides uniformity and equality in the final grade (see, e.g., [38, 69]).

Codingbat[5] generates a fixed set of tests for Java and Python and shows the student the result of (some of) these tests. Codingbat also keeps track of the student's history with a problem by producing a time-based graph showing each time the student tried to run the program and the percentage of unit tests that were successful.

CLoudCoder[6] [76] is still another open source web-based integrated development environment (IDE) for creating, assigning, and sharing short programming exercises for C/C++, Java, Python and Ruby. It is a property of Microsoft Research inspired by CodingBat. Similarly to CodingBat, CloudCoder uses unit tests in a web-based game to give players an opportunity to exercise their inductive skills. Players are given the results of a set of unit tests (calculated result versus expected result) and are asked to write code that will pass all the unit tests without knowing the problem description.

Code Hunt[7] is another system made by Microsoft Research [91]. An exercise requires the student to write a small amount of code as an answer to a given problem. The student is presented a series of puzzles that he/she has to explore through clues that are presented as test cases. The student must find a solution to the problem, introducing changes in the code that he/she writes until it matches the functional behavior of some secret solutions.

The correctness of the student's code is judged automatically by running the code against a set of tests. If all of the tests pass, the student's code is judged to be correct. Since this tool is web-based and the assessment is automatic, students can work

[4] http://coderunner.org.nz. Accessed 8 January 2021.

[5] http://codingbat.com. Accessed 8 January 2021.

[6] https://cloudcoder.org/. Accessed 8 January 2021.

[7] http://codehunt.com. Accessed 8 January 2021.

wherever and whenever is convenient for them, and receive immediate feedback (see, e.g., [87]).

Spinoza[8] [30] is Python IDE to enable classroom orchestration in introductory programming classes. It supports active learning [11] in introductory Java programming courses.

Active learning is any activity that is course-related and that all students in a class session are called upon to do other than simply watching, listening, and taking notes [46]. It is learner-centered, not teacher-centered; the active participation of a student is a necessary aspect in active learning. Students must be doing things and simultaneously think about the work done and the purpose behind it so that they can enhance their higher order thinking capabilities. The motivation behind this approach is that undergraduate students in classes with traditional stand-and-deliver lectures are 1.5 times more likely to fail than students in classes that use more stimulating active learning methods [49].

Spinoza includes the features of Codingbat, CloudCoder and Code Hunt. Spinoza differs from other similar systems in providing the instructor with detailed feedback in real time about the progress of the students during in-class coding sessions, both at the level of the individual student and the class as a whole. From the students' view, Spinoza provides authenticated access to a collection of named programming problems and a web-based IDE which allows students to code, run, unit test, and debug those problems. From the instructor's view, Spinoza provides interfaces for creating a variety of programming challenges and also provides multiple views of students' progress in solving the coding problems [31].

Spinoza can be used to generate this type of challenge but it does not provide the game-like interface of Code Hunt. Code Hunt, in turn, lacks the dashboard view of Spinoza.

Table 12.1 gives a qualitative evaluation of the studied objective tests in terms of (high, medium or low) ease of gaming the system by the student, the time for preparation of the test by the teacher, complexity of elaboration, depth of the tested knowledge, the field specificity, test efficiency, authenticity of the test, accessibility of the test, and the dependency on the statements' accuracy. Besides, a great overview of automatic assessment systems for programming until 2005 can be found in Douce et al. [36].

12.3.2 Assembly of Automated Objective Tests

The previously presented objective tests can be both grouped in a question (item) bank or a subpool in a Learning Management System (e.g., Moodle) and can be delivered online. The questions in a test may be selected manually or automatically (randomly) (e.g., one question per subpool), statically offline. There is a perception among students that randomly selected questions from item banks are unfair [35].

[8] https://github.com/abudeebf/Spinoza-2.0. Accessed 8 January 2021.

Table 12.1 Qualitative comparison of objective tests in terms of their characteristics

	Method					
Characteristic	MCQ	True/false	MQ	EMQ	SRQ	CTS
Gaming the system	High	High	Medium	Low	Low	Low
Time consuming for user	Low	Low	High	High	Medium	High
Complexity of elaboration	Low	Low	Medium	High	High	High
Depth of the tested knowledge	Low	Low	Low	High	Medium	High
Field specific	Low	Low	Low	High	Low	High
Test efficiency	Medium	Low	Low	High	High	High
Authenticity of the test	Low	Low	Low	High	Medium	High
Accessibility of the test	Medium	Medium	High	Low	High	Low
Dependent of the statements' accuracy	Low	Low	Low	High	High	Medium

The questions can be selected also dynamically as the examinee performs during the administration of the test (adaptive tests). Questions here are selected dynamically at testing time and depend on the prior performance of the examinee. (see, e.g., [68]).

Social networks as well as fast and simple group messaging and calling applications provide for easy communication among examinees, and therefore, facilitate academically dishonest behavior and cheating. Various high-security solutions exists to minimize and eliminate cheating during the exam (e.g. lockdown features in the OS, Safe Exam Browser, Windows Take a test) but it can be also decreased by conducting continuous background checks, logging and reporting of suspected cheating to the teacher (see, e.g., [37, 48, 52, 88]).

Student Learning Experience

Learning is enhanced through active involvement in personally meaningful experiences accompanied by processing for meaning and future use [70]. However, tests generally are an intimidating experience that should be neutralized by explaining to the students the purpose for the test and stress the positive effects it will have. Many may have very negative feelings left over from previous bad experiences.

Detailed planning and development of e-assessment can contribute to generalizing and increasing the quality of student learning experience, increasing the confidence in test levels of engagement and cooperation (see, e.g., [35]).

12.4 Bias in Objective Tests, Ethical Issues, Explainability

Artificial intelligence is sometimes defined as the discipline concerned with automating tasks that require intelligence [12]. Automation, in turn, is the process of enabling machines and software to operate without direct control or supervision

of people. When the outcome of automation has a direct impact on the well-being of people, we are rightly concerned with ensuring our capacity to adeptly monitor and analyze the automated activity. This concern has recently given rise to the field of AI ethics which includes, but is not limited to, the problems of accountability, transparency, fairness, and ethics.

Accountability

Accountability is defined in the Oxford dictionary as "the fact of being responsible for your decisions or actions and expected to explain them when you are asked" [1]. In Test-based accountability, tests are used to determine student status with respect to specific content standards. Every participant of the learning process is considered accountable for student test scores based on which the educational system has assigned criteria for rewards and sanctions for students, teachers, and/or schools. The educational system is responsible for communicating their functioning to educators and the public [59].

Test-based accountability can make teachers cover more material in the course and adopt better curricula or more effective teaching methods. However, test-based accountability can also make teachers teach only tested aspects incidental to the domain the test is intended to represent at the expense of untested aspects of the standards.

Intensified test-based accountability can also facilitate cheating on tests while decreasing equity, well-being, and justice (see, e.g., [81, 84]).

Another definition of accountability by Bovens [17] is:

> a relationship between an actor and a forum, in which the actor has an obligation to explain and to justify his or her conduct, the forum can pose questions and pass judgment, and the actor may face consequences.

In the context of AI and algorithms, we define algorithmic accountability as the accountability relationship in which the subject of the account is an algorithm [96].

When we are talking about the accountability of a system that is deployed, we are asking how we can empower a forum to attain accountability from the system. In contrast, when we are talking about a system in the design phase, then we are concerned with identifying who should have the power to change the behavior of the system being developed. When the new system is to replace human activity with an algorithm, we need to be concerned with the impact this change will have on powers that the user has. A human user can always freely and directly communicate with a human exam evaluator. However, this power is necessarily taken away when an automated evaluator is used. While we may not want to give students the power to change the behavior of an automated evaluator, we need to ensure a clear feedback communication path. This means that a student can react to perceived malfunction and should be given a timely response to the concern.

Transparency

Transparency is defined as the quantity and quality of information made available to the forum [33]. Test transparency is related with the clarity of information given to

the students regarding how they will be assessed and based on what criteria, e.g., how to answer the tests, available test time, the assessment procedure, and the formula for calculating the final grade. Test transparency also includes accessibility to sample questions and clarifying the evaluation criteria by making available to the students their detailed evaluation results.

The transparency of an algorithm is measured as the extent to which the available information about the algorithm that is subject to accountability allows the forum to monitor the workings or performance of that algorithm. In the context of automated evaluation, transparency can mean how much is the student aware of the workings of the evaluation system. This can be as trivial as the student does (not) know the evaluation is automated. It can also be that students are given details of the principles of the algorithm used; for example, whether a rule-based system is used, or a machine learning system, on which data is the system trained etc.

Explainability

Explainability, also called explainable AI (XAI), is explicitly concerned with the account, the possibility of generating it by human or automatic means [56]. When a decision or an evaluation is made about a person, particularly an undesirable one, explainability plays a crucial role to empower that person to improve their standing by mitigating the circumstances that led to the undesirable outcome. Beyond that explanation plays a role in enabling the designers of an algorithm to analyze its behavior particularly in specific contexts.

In the evaluator-student accountability relationship, explainability is a very important property of an automated system as it can make a difference to the learning experience. There are two aspects to be explained here. The first is shared with human evaluates: why is a particular answer marked as wrong? Explainability of AI and algorithms in general refers to its ability to identify how the input affects the output. In other words, an algorithm is explainable if it can provide information on how the outcome can be changed. There are three basic ways to attain explainability, but not all the three ways are possible for all types of algorithms.

The first is *explainability by design.* Some algorithms can be transparent to the user who can directly inspect them. For instance, informing a student they have obtained 125 points and that the threshold for a B is 130 points explains why they got a C. In the case of a multiple-choice exam, together with the solution key, this is sufficient explanation because the student learns how to improve their grade.

The second approach is "*backtracking*". Explainability can be accomplished by an algorithm that "backtracks" its own work to extract justifications or reasons for each decision it takes. This approach is particularly suited for rule-based systems. But tracking back the rules that have yielded the final evaluation, and which conditions were satisfied that made the rule applicable, one can understand why a particular evaluation was made. Knowing which rules the system can follow and changing some of the conditions, will result in an improved evaluation.

Lastly, there is the approach of *interpretation.* Explainability can also be accomplished by a second algorithm that creates interpretations of the decisions made by the algorithm that is subject of the account. This approach is explored for some machine

learning algorithms [56] where none of the other two approaches can be applied. In automated evaluation that uses machine learning, one has to be clear that the rule "learned" by training is not an inference but a correlation that heavily depends on the data used in training. The use of historic data in assigning grades comes with serious possible consequences[9] and if machine learning cannot be avoided, then it is very important to couple such an algorithm with strong accountability to the students and their guardians.

Fairness

When the output of an algorithm is different for different people, a major concern is whether that algorithm is *fair*, namely whether it works equally well for all persons. Fairness studies are motivated by the use of machine learning in the context of decision-making [77], however concerns for fairness extend to all algorithms that are part of systems that ethically impact individuals and society.

As an example of algorithmic bias in education, consider the case of a "virtual teaching assistant for a Georgia Tech course on artificial intelligence" which responded more successfully to concerns of male than those of female students [44]. The use of historical data can reinforce existing bias by transforming past correlations between socio-demographic properties and grades into present rules of inference.

The *unfairness* of an algorithm can be introduced at two points: by using input that has some kind of bias, or by having unfairness built in its operation and use. The ethical impact of an unfair algorithm is in an output that has an impact on equity, namely unevenly distributing resources among groups or individuals, but also by supporting or exalting existing undesirable phenomena in society. These points of introducing unfairness are not clearly separated and often bump into each other. *Algorithmic bias* refers to the second type of introduced unfairness: "Algorithmic bias is added by the algorithm itself and not present in the input data" [8]. Real-world examples of the occurrences of data bias are listed in, for example [62, 75].

Two general notions of fairness are considered, both of interest for decision-making and evaluation algorithms used in education: *group fairness* and *individual fairness*. In group fairness [75], we are concerned with statistical notions of fairness where a small number of protected demographic groups are identified. We are concerned with ensuring that (approximate) parity of some statistical measure across all these groups is maintained.

Individual fairness, on the other hand, is a metric that compares the similarity of evaluations for similar answers. In individual fairness [41], we are concerned that similar individuals should be treated similarly. A particular challenge for individual similarity is the identification of a metric for comparing the likeness of individuals. In group fairness we are concerned with the ability to identify many different metrics which may be mutually unaligned.

Fairness in machine learning is challenging to achieve for a variety of reasons [65]. Regardless of the problem, there sometimes does not exist a solution which is universally fair, equitable and just. The additional requirement of fairness can

[9] https://www.nytimes.com/2020/08/20/world/europe/uk-england-grading-algorithm.html.

come at a cost of efficiency—an algorithm can only be made fairer if it is made less correct. It has been shown that certain fairness types cannot be simultaneously achieved in machine learning [26]. In such situations, a trade-off would be necessary, meaning that we need to have a way to resolve such conflicts fairly. Both the data bias and algorithmic fairness measures evaluate a particular machine learning problem. However, local decisions, particularly when repeated over time, can lead to emerging global effects. Namely, by allowing a particular data bias to go undetected or by invisibly but consistently preferring one type of users over another, we may change our society. The risk of emerging effects from automated decision-making is difficult to localize, measure and mitigate.

The highest concern when we consider the fairness of algorithms that automate grading is that, as Chouldechova and Roth argue, "statistical definitions of fairness do not on their own give meaningful guarantees to individuals or structured subgroups of the protected demographic groups. Instead, they give guarantees to 'average' members of the protected groups" [27].

When the new system is to replace human activity with an algorithm, we need to be concerned with the impact this change will have on the service provider-service recipient relationship. In this relationship, both parties have both rights and obligations. One must be vigilant as to how this relationship changes when the evaluator role is supplemented with an algorithm. A somewhat neglected aspect of fairness is the disruption of this power balance: more obligations are transferred from the human evaluator to the human student. For example, in non-multiple-choice tests, a human evaluator will directly comment on the answer and how it can be improved. The student gets this comment when receiving the evaluation. When that same evaluation is automated, the answer can be marked as incorrect, but a human evaluator needs to supply the explanation. Now, the student would have to explicitly ask for the explanation, thus introducing an added obligation on the student that was not there before: figure out how to ask for the explanation and from whom. This too can be considered unfair behavior: students that have a strong technical understanding of the system will be at an advantage.

Privacy

Lastly, the much-discussed property of algorithms today is that of *privacy*. As discussed in Garshi et al. [51], new technology offers to blur the line between supervision and surveillance and change the learning experience altogether. Privacy in the context of performance evaluation is not a concern that is unique to new technologies. There is no one definition of what privacy or breach of privacy is but as per Solove [86], privacy can be breached by three main information-related activities: information collection, processing, and dissemination.

In the context of automated evaluation, this means a student's privacy can be violated when data from the student is collected, when the evaluation is done and when the results of the evaluation are disseminated. We would add that the world of machine learning algorithms offers a fourth possibility—including information from the evaluation process in a data set further used in training algorithms or made available to third parties. The topic of the "classic" three privacy vulnerable points

is, as far as we are aware, not explored in the context of automated grading and evaluation.

The problem of student data use is a pertinent one that is to a certain extent tackled by ensuring that any data collected satisfies differential privacy conditions [40]. This means that data points that would make it possible for a student to be identified in a data collection are removed. While ensuring differential privacy is the first step in guaranteeing the privacy of students whose data is collected, it is not a magic bullet. The problem of specifying what constitutes a violation of privacy in automated grading and evaluation and how to prevent those violations is still an open challenge.

12.5 Comparison of Factual Tests and Code Testing Systems

This section discusses the selection of an appropriate type of test for each type of knowledge: understanding of programming concepts and principles that do not include writing program code, application—simple programming tasks, analysis—debugging tasks, and synthesis—advanced programming assignments. We compare the five principles of testing as well as ethical issues and explainability of the previously presented factual tests including multiple choice questions, true/false questions, matching questions, extended matching questions, and short response questions as well as code testing systems including Web-Cat, CodeRunner, Codingbat, Cloud-Coder, Code Hunt, and Spinoza.

The performance indicators we consider are learning experience, transparency, explainability, authenticity, reliability, practicality, and positive washback, Table 12.2.

The rating system of the studied performance indicators is represented by a 3-point rating scale that measures whether a goal was accomplished. A 3 ranking implies that a goal was met, a 2 ranking is given to partially met goals, and a 1 ranking is assigned to an unfinished goal where most or all dimensions were not achieved. Note that the indicated rating is given for the optimized questions designed to respond at best to the type of the question.

A pleasant experience with an online test does not stress the student unnecessarily and it facilitates and stimulates learning. Learning experience in factual tests is generally much lower than in code testing systems whose user interface (UI) plays a major role that is generally useful, with an intuitive interface and the structure and presentation of content and multimedia, easy to navigate, with a good graphic design for a high-quality learning experience.

In the case of ambiguity of a factual test, the lack of space prohibits the students to explain their answer and the interpretation of the question, which can result in a negative experience. Moreover, code testing systems run in iterations. Spinoza is apt for active learning, but it does not provide the game-like interface of CodeHunt.

Table 12.2 Comparison of factual tests and code testing systems considering learning experience, transparency, explainability, reliability, practicality, and positive washback

Group →	Factual tests						Code testing systems				
	Method										
Category	MCQ	True/false	MQ	EMQ	SRQ	Web-Cat	CodeRunner	Codingbat	CloudCoder	CodeHunt	Spimoza
Learn. experience?	2	1	2	1	2	3	3	3	3	3	3
Transparent?	3	3	3	3	3	2	2	2	2	2	1
Explaninable?	3	3	3	3	3	3	3	3	3	3	3
Authentic?	2	1	1	1	1	3	3	3	3	3	3
Reliable?	3	3	3	3	3	2	2	2	2	2	2
Pratical?	3	3	3	3	3	2	2	2	2	2	2
Pos. washback?	3	3	3	3	2	2	2	2	2	2	2

The transparency of factual texts in general is higher than the transparency of code testing systems since the rules of evaluation are very simple and given in advance. In the studied code testing systems, in static analysis, the students' code is compared with the model programs given by a teacher: the closer the student code is to the model programs, the higher is the mark. Here it is more difficult to give a clear-cut explanation on the grade as there are more sources of error that may be syntactic and structural. Optionally, manual grading by the teacher with direct on-line markup of code is possible, which is a type of subjective grading prone to bias and to other errors discussed previously.

All the studied factual tests and code testing systems are explainable since we can interpret the marks given while being able to clearly traverse back, from the grades to the student answers, on the rule-based path the test took to arrive at the marks.

The authenticity of factual tests in evaluating computer programming knowledge and skills is generally much lower than the authenticity of code testing systems.

Reliability in Table 12.2 refers to test reliability that can be improved by clear task instructions, clear assessment criteria, and scales. However, creating an analytic scale for marking a test is more difficult in the case of code testing systems than in factual tests.

In general, practicality in factual tests is also higher than in code testing systems since the time and effort that the development of models and feedback delivery require in code testing systems is much higher than in factual tests. Moreover, both in factual tests as in code testing systems, it is difficult to give a personalized positive washback that helps students to improve their skills. While feedback in code testing systems mostly focuses on identifying errors, factual tests let the teacher adapt washback for each type of answer. However, detecting errors in code-testing helps the student to fix them and thus evolve in an iterative code development, even though not in the most constructive environment.

Since validity depends on the type of knowledge tested, we compare the latter in previously mentioned tests for testing basic concepts and paradigms, simple programming, debugging, and advanced programming. Additionally, we give the rating of the matching of the four learning objective categories in computer science courses with the discussed factual tests and code testing systems aiming at providing a higher degree of test validity in Table 12.3. The 3-point rating scale that measures whether a goal was accomplished here is the same as in Table 12.2.

The validity of factual tests in testing basic computer science concepts and paradigms is high, contrary to code testing systems since here we speak about understanding the basic constructive concepts, definitions, and their relations in computer science. However, even though we may design multiple-choice questions for testing simple programming skills, factual tests are, in general, not apt for testing the knowledge and skills in simple programming, debugging, and advanced programming. Additionally, the studied code-testing systems are apt for testing introductory programming knowledge and skills but are not suitable for advanced programming evaluation for which code testing systems are still waiting to be developed.

Table 12.3 Validity comparison of factual tests and code testing systems

Group →	Factual tests						Code testing systems				
	Method										
Category	MCQ	True/false	MQ	EMQ	SRQ	Web-Cat	CodeRunner	Codingbat	CloudCoder	CodeHunt	Spimoza
Basic concepts	3	3	3	3	3	1	1	1	1	1	1
Simple programming	2	2	2	2	2	2	3	3	3	3	3
Debugging	1	1	1	1	1	3	3	3	3	3	3
Adv. programming	1	1	1	1	1	2	2	2	2	2	2

12.6 Conclusions and Future Challenges

To achieve creative and critical thinking in our students and to facilitate their collaboration and lifelong learning given the constant evolution of technology and the related high probability of changing career paths, we need to provide for a fast development of new skills and the change in the assessment going beyond conventional reactive rule-based automated grading systems.

Contrary to traditional lecturing mostly used in the classroom, active learning through online learning technologies (e.g., Spinoza) is a preferable learning approach that specifically presupposes students' activity. It benefits on average all students while offering increased benefits for students from underrepresented groups. Widespread implementation of high-quality active learning can help reduce learning gaps in computer science courses and promote student equity (see, e.g., [90]).

In Table 12.4, we give a three-level qualitative grading scale comparison (more, equal, and less) of software based objective tests versus subjective tests performed by human evaluators in terms of the following characteristics: fairness, bias, transparency, explainability, and reliability. If designed well, in general, objective tests may be more fair, less biased, more transparent, explainable, and more reliable compared to subjective tests performed by human evaluators. This summative and coarse comparison should be taken with reserve due to significant variations in terms of subdiscipline and other relevant parameters that influence the choice of the test. Since the discipline of learning assessment develops rapidly, it is likely that this qualitative comparison will change in the future.

Yet another issue with advanced programming assessment tools in online tests is that they still depend on human teacher evaluation capacity that cannot yet be easily automatized. Thus, the scalability of these tools depends on the availability of skilled teaching staff and on the level of preparedness and the homogeneity of the academic background of the students in the class.

To bridge the gap between the active learning requirements for automated grading of computer science courses online and the state-of-the-art online grading methods, automated grading should consider learning preferences and constraints of each student. Instead of reactive rule-based objective tests, a proactive automated grading system of the future should be able to function as a learning assistant adaptable to the learning pace, characteristics, preferences, and abilities of each student whether they are at introductory, intermediate, or advanced programming level. Such a system should also serve for connecting compatible students in creating study groups in large MOOCs and thus facilitate students' empowerment and learning autonomy, where seamless meeting of new peers and establishing study networks among students

Table 12.4 Qualitative comparison of automated (software based) objective tests versus subjective (human evaluated) tests

Fairness	Bias	Transparency	Explainability	Reliability
More	Less	More	More	More

is still an open challenge. From our perspective, software agent technology seems to be a promising line of research for active learning and efficient and effective application of code testing systems in advanced programming based on the learning pace, requirements and needs of each student.

References

1. Accountability, Oxford learner's dictionaries (2021), https://www.oxfordlearnersdictionaries.com/definition/english/accountability. Accessed 9 Jan 2021
2. P.W. Airasian, *Classroom Assessment: Concepts and Applications* (ERIC, 2001)
3. M.O. Al-Rukban, Guidelines for the construction of multiple choice questions tests. J. Family Commun. Med. **13**(3), 125 (2006)
4. K.M. Ala-Mutka, A survey of automated assessment approaches for programming assignments. Comput. Sci. Educ. **15**(2), 83–102 (2005)
5. J. Anderson, Sex-related differences on objective tests among undergraduates. Educ. Stud. Math. **20**(2), 165–177 (1989)
6. L.W. Anderson, B.S. Bloom et al., *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives* (Longman, 2001)
7. J.M.M.L. Azevedo, A.P. Lopes, M.D.L. Babo, C. Torres, Multiple-choice tests: a tool in assessing knowledge. ICERI (2010)
8. R. Baeza-Yates, Bias on the web. Commun. ACM **61**(6), 54–61 (2018). https://doi.org/10.1145/3209581
9. S.P. Balfour, Assessing writing in moocs: Automated essay scoring and calibrated peer review™. Res. Pract. Assess. **8**, 40–48 (2013)
10. K. Beck, *Test-Driven Development: By Example* (Addison-Wesley Professional, 2003)
11. B.S. Bell, S.W. Kozlowski, Active learning: effects of core training design elements on self-regulatory processes, learning, and adaptability. J. Appl. Psychol. **93**(2), 296 (2008)
12. R.E. Bellman, *An Introduction to Artificial Intelligence: Can Computers Think?* (Boyd & Fraser Publishing Company, 1978)
13. G. Ben-Shakhar, Y. Sinai, Gender differences in multiple-choice tests: the role of differential guessing tendencies. J. Educ. Measur. **28**(1), 23–35 (1991)
14. R. Bennett, M. Goodman, J. Hessinger, H. Kahn, J. Ligget, G. Marshall, J. Zack, Using multimedia in large-scale computer-based testing programs. Comput. Hum. Behav. **15**(3–4), 283–294 (1999)
15. R.E. Bennett, The changing nature of educational assessment. Rev. Res. Educ. **39**(1), 370–407 (2015)
16. B.S. Bloom, M.D. Engelhart, E.J. Furst, W.H. Hill, D.R. Krathwohl, Taxonomy of educational objectives: the classification of educational goals. *Handbook I Cognitive Domain* (David McKay Company Inc, New York, 1956)
17. M. Bovens, Analysing and assessing accountability: a conceptual framework1. Eur. Law J. **13**(4), 447–468 (2007). https://doi.org/10.1111/j.1468-0386.2007.00378.x
18. J.A. Brabec, S.C. Pan, E.L. Bjork, R.A. Bjork, True-false testing on trial: guilty as charged or falsely accused? Educ. Psychol. Rev. 1–26 (2020)
19. G.A. Brown, J. Bull, M. Pendlebury, *Assessing Student Learning in Higher Education* (Routledge, 2013)
20. H.D. Brown, P. Abeywickrama, *Language Assessment: Principles and Classroom Practices*, vol. 10 (Pearson Education White Plains, NY, 2010)
21. J.C. Caiza, J.M. Del Alamo, Programming assignments automatic grading: review of tools and implementations, in *7th International Technology, Education and Development Conference (INTED2013)* (2013), p. 5691

22. B. Canou, R.D. Cosmo, G. Henry, Scaling up functional programming education: under the hood of the Ocaml MOOC. Proc. ACM Program. Lang. **1**(ICFP):4:1–4:25 (2017). https://doi.org/10.1145/3110248
23. Y. Cao, L. Porter, S.N. Liao, R. Ord, Paper or online? A comparison of exam grading techniques, in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (2019), pp. 99–104
24. E.G. Carmines, R.A. Zeller, *Reliability and Validity Assessment* (Sage Publica tions, 1979)
25. G. Charness, U. Gneezy, Strong evidence for gender differences in risk taking. J. Econ. Behav. Organ. **83**(1), 50–58 (2012)
26. A. Chouldechova, Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. Big Data **5**(2), 153–163 (2017). https://doi.org/10.1089/big.2016.0047,doi:10.1089/big.2016.0047
27. A. Chouldechova, A. Roth, A snapshot of the frontiers of fairness in machine learning. Commun. ACM **63**(5), 82–89 (2020). https://doi.org/10.1145/3376898
28. J. Clune, V. Ramamurthy, R. Martins, U.A. Acar, Program equivalence for assisted grading of functional programs, vol. 4 (OOPSLA, 2020). https://doi.org/10.1145/3428239
29. I. Dabbebi, S. Iksal, J.M. Gilliot, M. May, S. Garlatti, Towards adaptive dashboards for learning analytic: an approach for conceptual design and implementation, in *9th International Conference on Computer Supported Education (CSEDU 2017)*, Porto, Portugal (2017), pp. 120–131. https://doi.org/10.5220/0006325601200131
30. F.A. Deeb, T. Hickey, Spinoza: the code tutor, in *Proceedings of the International Conference on Computer and Information Science and Technology*, Ottawa, Canada (2015)
31. F.A. Deeb, T. Hickey, Flipping introductory programming classes using spinoza and agile pedagogy, in *2017 IEEE Frontiers in Education Conference (FIE)* (IEEE, 2017), pp. 1–9
32. P. Denny, S. Manoharan, U. Speidel, G. Russello, A. Chang, On the fairness of multiple-variant multiple-choice examinations, in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), pp. 462–468
33. N. Diakopoulos, Transparency, in *The Oxford Handbook of Ethics of AI*, ed. M.D. Dubber, F. Pasquale, S. Das (Oxford University Press, 2020). https://doi.org/10.1093/oxfordhb/9780190067397.013.11
34. A. Dimitrakopoulou, State of the art on interaction and collaboration analysis. (D26.1.1) EU Sixth Framework programme priority 2, Information society technology, Network of Excellence Kaleidoscope, (contract NoE IST-507838), project ICALTS: Interaction & Collaboration Analysis (2004)
35. G. Domino, M.L. Domino, *Psychological Testing: An Introduction* (Cambridge University Press, 2006)
36. C. Douce, D. Livingstone, J. Orwell, Automatic test-based assessment of programming: a review. J. Educ. Resourc. Comput. (JERIC) **5**(3), 4–es (2005)
37. F. Drasgow, *Technology and Testing: Improving Educational and Psychological Measurement* (Routledge, 2015)
38. W. Du, *Code Runner: Solution for Recognition and Execution of Handwritten Code* (Standford University, 2012), pp. 1–5
39. E. Duval, M. Sharples, R. Sutherland, *Technology Enhanced Learning* (Springer, 2017)
40. C. Dwork,F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in *Theory of Cryptography*, ed. S. Halevi, T. Rabin (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 265–284
41. C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R.S. Zemel, Fairness through awareness (2011), http://arxiv.org/abs/1104.3913
42. R.L. Ebel, D.A. Frisbie, *Essentials of Educational Measurement* (Prentice-Hall Englewood Cliffs, NJ, 1972)
43. S.H. Edwards, M.A. Perez-Quinones, Web-cat: automatically grading programming assignments, in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (2008), pp. 328–328

44. B. Eicher, L. Polepeddi, A. Goel, Jill Watson doesn't care if you're pregnant: grounding AI ethics in empirical studies, in: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, Association for Computing Machinery*, New York, NY, USA, AIES '18 (2018), pp. 88–94. https://doi.org/10.1145/3278721.3278760
45. Elias T (2011) Learning analytics. Learning 1–22
46. R.M. Felder, R. Brent, Active learning: an introduction. ASQ High. Educ. Brief **2**(4), 1–5 (2009)
47. R. Ferguson, Learning analytics: drivers, developments and challenges. Int. J. Technol. Enhanc. Learn. **4**(5–6), 304–317 (2012)
48. M. Finkelman, M.L. Nering, L.A. Roussos, A conditional exposure control method for multidimensional adaptive testing. J. Educ. Measur. **46**(1), 84–103 (2009)
49. S. Freeman, S.L. Eddy, M. McDonough et al., Active learning increases student performance in science, engineering, and mathematics. Proc. Natl. Acad. Sci. **111**(23), 8410–8415 (2014)
50. S. García-Molina, C. Alario-Hoyos, P.M. Moreno-Marcos, P.J. Muñoz-Merino, I. Estévez-Ayres, C. Delgado Kloos, An algorithm and a tool for the automatic grading of MOOC learners from their contributions in the discussion forum. Appl. Sci. **11**(1) (2021). https://doi.org/10.3390/app11010095, https://www.mdpi.com/2076-3417/11/1/95
51. A. Garshi, M.W. Jakobsen, J. Nyborg-Christensen, D. Ostnes, M. Ovchinnikova, M. Slavkovik, Smart technology in the classroom: systematic review and prospects for algorithmic accountability, in *Handbook of Intelligent Techniques in Educational Process* (Springer, in this book, 2021)
52. E. Georgiadou, E. Triantafillou, A.A. Economides, A review of item exposure control strategies for computerized adaptive testing developed from 1983 to 2005. J. Technol. Learn. Assess. **5**(8), n8 (2007)
53. A. Gopal, Internationalization of higher education: preparing faculty to teach cross-culturally. Int. J. Teach. Learn. High. Educ. **23**(3), 373–381 (2011)
54. C. Gordon, J. Hughes, C. McKenna, Assessment toolkit ii: Time-constrained examinations (2017), https://london.ac.uk/sites/default/files/cde/assessment-toolkit-II-2017.pdf. Accessed 9 Jan 2021
55. N.E. Gronlund, *Assessment of Student Achievement* (ERIC, 1998)
56. D. Gunning, D. Aha, Darpa's explainable artificial intelligence (XAI) program. AI Mag. **40**(2), 44–58 (2019)
57. T.M. Haladyna, *Developing and Validating Multiple-Choice Test Items* (Rout ledge, 2004)
58. R.K. Hambleton, H. Swaminathan, *Item Response Theory: Principles and Appli cations* (Springer Science & Business Media, 2013)
59. L.S. Hamilton, B.M. Stecher, S.P. Klein, *Making Sense of Test-Based Accountability in Education* (Rand Corporation, 2002)
60. Y. Han, W. Wu, Y. Yan, L. Zhang, Human-machine hybrid peer grading in spocs. IEEE Access **8**, 220922–220934 (2020). https://doi.org/10.1109/ACCESS.2020.3043291
61. R.M. Harden, Learning outcomes and instructional objectives: is there a difference? Med. Teach. **24**(2), 151–155 (2002)
62. M. Kearns, A. Roth, *The Ethical Algorithm: The Science of Socially Aware Algorithm Design* (Oxford University Press, 2019)
63. H. Keuning, J. Jeuring, B. Heeren, A systematic literature review of automated feedback generation for programming exercises. ACM Trans. Comput. Educ. (TOCE) **19**(1), 1–43 (2018)
64. R. Killen, Validity in outcomes-based assessment. Perspect. Educ. **21**(1), 1–14 (2003)
65. M.J. Kusner, J.R. Loftus, The long road to fairer algorithms. Nature 34–36 (2020). https://doi.org/10.1038/d41586-020-00274-3
66. A. Lajis, S.A. Baharudin, D. Ab Kadir, N.M. Ralim, H.M. Nasir, N.A. Aziz, A review of techniques in automatic programming assessment for practical skill test. J. Telecommun. Electron. Comput. Eng. (JTEC) **10**(2–5), 109–113 (2018)

67. A. Lebis, M. Lefevre, V. Luengo, N. Guin, Capitalisation of analysis processes: enabling reproducibility, openess and adaptability thanks to narration, in *LAK '18—8th International Conference on Learning Analytics and Knowledge* (ACM, Sydney, Australia, 2018), pp. 245–254. https://doi.org/10.1145/3170358.3170408, https://hal.archives-ouvertes.fr/hal-01714184
68. W.J. van der Linden, C.A. Glas, *Elements of adaptive testing* (Springer, 2010)
69. R. Lobb, J. Harlow, Coderunner: a tool for assessing computer programming skills. ACM Inroads **7**(1), 47–51 (2016)
70. Luckner JL, Nadler RS (1997) Processing the experience: Strategies to enhance and generalize learning. ERIC
71. J.M. Malouff, E.B. Thorsteinsson, Bias in grading: a meta-analysis of experimental research findings. Aust. J. Educ. **60**(3), 245–256 (2016)
72. H.W. Marsh, Students' evaluations of university teaching: dimensionality, reliability, validity, potential biases and usefulness, in *The Scholarship of Teaching and Learning in Higher Education: An Evidence-Based Perspective* (Springer, 2007), pp. 319–383
73. P. McCoubrie, Improving the fairness of multiple-choice questions: a literature review. Med. Teach. **26**(8), 709–712 (2004)
74. S.E. Meek, L. Blakemore, L. Marks, Is peer review an appropriate form of assessment in a MOOC? Student participation and performance in formative peer review. Assess. Eval. High. Educ. **42**(6), 1000–1013 (2017)
75. N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning (2019). CoRR http://arxiv.org/abs/1908.09635
76. A. Papancea, J. Spacco, D. Hovemeyer, An open platform for managing short programming exercises, in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (2013), pp. 47–52
77. D. Pessach, E. Shmueli, Algorithmic fairness. 2001.09784 (2020)
78. J. Pivarski, C. Bennett, R.L. Grossman, Deploying analytics with the portable format for analytics (PFA), in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016), pp. 579–588
79. Y. Reyneke, C.C. Shuttleworth, R.G. Visagie, Pivot to online in a post-covid-19 world: critically applying BSCS 5E to enhance plagiarism awareness of accounting students. Account. Educ. 1–21 (2020)
80. E.G. Rizkallah, V. Seitz, Understanding student motivation: a key to retention in higher education. Sci. Ann. Econ. Bus. **64**(1), 45–57 (2017)
81. J. Ro, Learning to teach in the era of test-based accountability: a review of research. Prof. Dev. Educ. **45**(1), 87–101 (2019)
82. R. Romli, S. Sulaiman, K.Z. Zamli, Automatic programming assessment and test data generation a review on its approaches, in *2010 International Symposium on Information Technology*, vol 3 (IEEE, 2010), pp. 1186–1192
83. C.A. Rowland, The effect of testing versus restudy on retention: a meta-analytic review of the testing effect. Psychol. Bull. **140**(6), 1432 (2014)
84. D. Santori, Test-based accountability in England, in *Oxford Research Ency clopedia of Education* (2020)
85. N. Siddiquei, R. Khalid, The relationship between personality traits, learning styles and academic performance of e-learners. Open Praxis **10**(3), 249–263 (2018)
86. D.J. Solove, A taxonomy of privacy. Univ. Pennsylvania Law Rev. **154**(3), 477–564 (2006), http://www.jstor.org/stable/40041279
87. J. Spacco, P. Denny, B. Richards, D. Babcock, D. Hovemeyer, J. Moscola, R. Duvall, Analyzing student work patterns using programming exercise data, in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015), pp. 18–23
88. M.L. Stocking, C. Lewis, Methods of controlling the exposure of items in cat, in *Computerized Adaptive Testing: Theory and Practice* (Springer, 2000), pp. 163–182
89. L. Suskie, *Assessing Student Learning: A Common Sense Guide* (Wiley, 2018)
90. E.J. Theobald, M.J. Hill, E. Tran, et al., Active learning narrows achievement gaps for underrepresented students in undergraduate science, technology, engineering, and math. Proc. Natl. Acad. Sci. **117**(12), 6476–6483 (2020)

91. N. Tillmann, J. De Halleux, T. Xie, J. Bishop, Code hunt: gamifying teaching and learning of computer science at scale, in *Proceedings of the First ACM Conference on Learning@ Scale Conference* (2014), pp. 221–222
92. K.J. Topping, Peer assessment. Theory Pract. **48**(1), 20–27 (2009)
93. O.M. Ventista, Self-assessment in massive open online courses. E-Learn. Digit. Media **15**(4), 165–175 (2018)
94. E. Ventouras, D. Triantis, P. Tsiakas, C. Stergiopoulos, Comparison of examination methods based on multiple-choice questions and constructed-response questions using personal computers. Comput. Educ. **54**(2), 455–461 (2010)
95. K. Verbert, N. Manouselis, H. Drachsler, E. Duval, Dataset-driven research to support learning and knowledge analytics. J. Educ. Technol. Soc. **15**(3), 133–148 (2012)
96. M. Wieringa, What to account for when accounting for algorithms: a systematic literature review on algorithmic accountability, in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery*, New York, NY, USA, FAT*'20 (2020), pp. 1–18. https://doi.org/10.1145/3351095.3372833
97. G. Wiggins, Assessment: authenticity, context, and validity. Phi delta kappan **75**(3), 200–213 (1993)
98. E. Wood, What are extended matching sets questions? Biosci. Educ. **1**(1), 1–8 (2003)
99. C. Wyatt-Smith, J. Cumming, *Educational assessment in the 21st century* (Springer, 2009)
100. J. Xu, Q. Li, J. Liu, P. Lv, G. Yu, Leveraging cognitive diagnosis to improve peer assessment in MOOCS. IEEE Access **9**, 50466–50484 (2021). https://doi.org/10.1109/ACCESS.2021.3069055