Assessment for Computer Programming Courses: A Short Guide for the Undecided Teacher

João-Paulo Barros[1,2][a]
[1]*Polytechnic Institute of Beja, Beja, Portugal*
[2]*Centre of Technology and Systems-UNINOVA, Caparica, Portugal*

Keywords: Assessment, Grading, Computer Science Education, Programming, CS1, CS2.

Abstract: As the large number of articles on teaching introductory programming seem to attest, teaching and learning computer programming is difficult. However, perhaps surprisingly, the assessment design for those courses does not seem to be the most studied aspect. This short position paper provides a structured set of options and alternatives to consider when choosing the assessment elements for a programming course. The objective is to promote additional reflection on several alternatives for each assignment, exam, or other assessment elements. Along with this presentation, we point to eventually valuable references. We believe the resulting information should be helpful and applicable to many other disciplines, but the focus is on computer programming courses.

1 INTRODUCTION

Programming, including introductory programming, is widely regarded as an essential topic in computer science teaching, arguably the most important one. The vast number of scientific articles attests to this, probably due to the recognized difficulty in effectively teaching novices how to program a computer, e.g., (Bennedsen and Caspersen, 2007; Bennedsen and Caspersen, 2019; Luxton-Reilly et al., 2018; Medeiros et al., 2019; Salguero et al., 2021; Williams et al., 2021).

It seems evident that the assessment and grading of programming assignments, exams, and other assessment elements is an important related topic. However, perhaps surprisingly, an extensive survey of introductory programming articles found that assessment — more specifically assessment tools, approaches to assessment, feedback on assessment, and academic integrity — represents only 192 in a total of 1666 examined papers (Luxton-Reilly et al., 2018). The remaining ones were focused on "the student" (489), "teaching" (905), and the "curriculum" (258).

Although teaching practices and assessment activities are essential (Salguero et al., 2020), here we focus on the characteristics of assessment elements. Assessment is, undoubtedly, a critical topic as it is a fundamental component of course design to which learning and teaching activities should be aligned, e.g., (Biggs and Tang, 2011). Also, as there is no general agreement on how to teach programming or even the fundamental topics (Luxton-Reilly et al., 2018), it should be of no surprise that assessment is also a subject not only of permanent discussion and research but also of doubt and uncertainty. Additionally, the need to move courses to online or blended learning made assessment design even more complex by adding additional concerns, most notably an increased worry with plagiarism and authorship, e.g., (Garg and Goel, 2022). This paper proposes a short hierarchical guide to aid the programming course designer in selecting the desired properties for each assessment element in a programming course. The guide is applicable to many other courses; yet, all the presented possibilities result from the author's experience teaching programming courses and are considered in that context. We present the guide while pointing to some articles with additional information about assessment options, emphasizing computer programming courses, before concluding.

2 A GUIDE FOR DECISION MAKING

Assessment choices are vast. However, our objective is to provide a relatively short guide that is complete

[a] https://orcid.org/0000-0002-0097-9883

Barros, J.
Assessment for Computer Programming Courses: A Short Guide for the Undecided Teacher.
DOI: 10.5220/0011095800003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 2*, pages 549-554
ISBN: 978-989-758-562-3; ISSN: 2184-5026

enough to provide a valuable basis for reflecting and deciding on the course design, namely the assessment and its alignment with teaching and learning activities. It is important to note that any learning activity can be used as an assessment element, and any assessment element can be used as a learning activity. Nevertheless, our guide does not focus on a list of assessment activities. Area agnostic lists are readily available in book form, e.g., (Brown et al., 2004), as well as programming specific proposals, e.g., (Malmi and Korhonen, 2008; Nørmark et al., 2008). Instead, we focus on a set of characteristics that can be used to classify activities, namely those related to assessment and grading. The guide can thus be seen as a reminder about the main properties of any single activity the teacher intends to use as an assessment. However, the hypotheses are still huge as these properties can easily be combined for each assessment element, and a single course can easily include several assessment elements.

We present a hierarchical representation for our guide showing the relations between a significant set of properties that we can assign to any assessment element. Additional possible relations were left out to provide better readability. The assessment element can be any activity used as an assignment, e.g., written exam, lab exams, clickers in the classroom, short quizzes along the semester, or others.

The hierarchy is not a proper decision, or even a classification, tree, as one path from the root node to a leaf is not a complete hierarchical decision or classification. Instead, one should follow several paths from the root node to one leaf for each assessment element we intend to characterize or classify. Each of these top-down paths will provide one or more characteristics for that assessment element. For example, a traditional paper-based coding exam can be classified by the following set of paths:

1. location→ in-class;
2. question→ functional;
3. question→ written question;
4. question→ closed-book;
5. answer→ written→ code paper-based;
6. authorship→ individual;
7. assessment/grading→ analytic;
8. assessment/grading→ scale/n-ary;
9. assessment/grading→ teacher assessment;
10. assessment/grading→ quantitative;
11. assessment/grading→ additive;
12. assessment/grading→ weighted average;
13. assessment/grading→ minimal grade;
14. cardinality→ single.

Hence, the hierarchy allows the teacher to quickly visualize all the known options. It has the additional advantage of being easy for each teacher to enrich based on their knowledge and experience regarding assessment options. This can result in removing some options the teacher does not favor or, most probably, adding additional ones. An alternative graphical representation, e.g., a mind map, was omitted due to the resulting weak readability.

Next, we present the hierarchy by briefly discussing each of the five main branches: (1) authorship, (2) location, (3) question, (4) answer, (5) cardinality, and (6) assessment/grading. Along the way, we suggest some possibilities to enrich the hierarchy and present some pointers to potentially valuable articles. We use boldface for the first occurrence of each term in the guide.

1. Authorship
   (a) group
   (b) individual
2. Location
   (a) in-class
   (b) mixed
   (c) take-home (asynchronous)
   (d) remote (synchronous)
3. Question
   (a) Format
      i. written question
      ii. for multi-choice answer — (answer-written) multi-choice
      iii. oral question
   (b) Type
      i. functional — (answer-written) descriptive, multi-choice, paper-based, computer-based
      ii. non-functional — (answer-written) descriptive
   (c) Documentation
      i. closed-book
      ii. open-book
      iii. partially open-book
4. Answer
   (a) oral
   (b) written
      i. computer-based
      ii. paper-based
      iii. multi-choice
      iv. descriptive
5. Cardinality
   (a) single

(b) multiple; see assessment/grading mandatory (minimal grade), weighted average, improve previous grade, remove some (e.g., best or worst)

6. Assessment/grading
   (a) Type of grading
      i. norm-referenced (grading on the curve)
      ii. criterion-referenced (learning objectives)
      iii. analytic (or atomistic)
      iv. holistic (or global)
         A. relative order and feedback
         B. rubrics
   (b) Creditation
      i. mandatory
      ii. optional
         A. for normal credit
         B. for extra credit
      iii. additive (e.g., for an weighted average)
      iv. subtractive (penalization)
   (c) Total grade
      i. mandatory (minimal grade)
      ii. weighted average
      iii. improve previous grade
      iv. remove some (e.g., best or worst)
   (d) Human/Machine
      i. machine-based assessment — (answer-written) computer-based, multi-choice
      ii. human-based assessment
         A. peer-assessment
         B. teacher-assessment
   (e) Scale
      i. n-ary (qualitative (rubrics) or quantitative)
      ii. binary

2.1 Authorship: Individual or Group

Group assignments tend towards a climate of trust environment as it becomes more difficult to assign individual grades. Yet, they also provide additional opportunities for improved learning and obvious advantage to foster soft skills. Nevertheless, **individual** assessment in a trusted environment, namely take-home exercises, can provide a less stressful and productive environment for some students, most notably the ones in the autistic spectrum (Stuurman et al., 2019). In summary, it seems easy to conclude that a mix of both types of activities should be the preferred option.

2.2 Location: In-class, Home, Mixed, or Remote

Traditional assessment activities, most notably exams, are conducted **in-class** under teacher surveillance to reduce the opportunities for academic fraud. However, it is a well-known fact that a trust climate, sometimes named "Theory Y," is better for learning, while a "don't trust" climate ("Theory X") is better for administration purposes (e.g., (Biggs and Tang, 2011)). Therefore, a better option is to tend towards a trust climate, which should mean a significant part of **take-home** activities to foster learning, which means a **mixed** set of assessments.

A well-known way to fight plagiarism is to make some low-control graded items dependent on a high-control one. This was proposed at least as of 1982 (Hwang and Gibson, 1982).

We can also include **remote** assessments in the mix of in-class and take-home activities. It is especially useful for oral discussions and presentations when physical presence is not mandatory, possible, or convenient.

2.3 Question

Each assessment activity will have the students answer one or more **question**s (or requirements). Each can be stated as **oral question**s or **written question**s. When in written form, they should be carefully constructed accordingly to the answer type: **code computer-based**, **code paper-based**, **multi-choice answer**, or **descriptive**. It is not possible to overemphasize the importance of carefully formulating each question for the intended purposes. Probably, the only safe way to test it is to give it to other teachers or, if possible and even better, to other students. However, even a later double-check reading can be extremely valuable to verify the questions' correctness.

Two significant decisions should be made regarding the type of questions: if the intended requirements are **functional** (e.g., the code is required to do something) or **non-functional** (e.g., the program is supposed to comply with something); if the assessment is **closed-book**, **open-book**, or **partially open-book** (a cheat-sheet is allowed (de Raadt, 2012) or some other specific resources can be consulted).

Recently, due to the COVID-19 pandemic, the remote open-book variants became much more popular. Interestingly, this does not seem to imply increased fraud: in one study, no statistically significant differences were reported comparing online open-book vs. traditional closed-book exams (Quille et al., 2021).

Naturally, some question types are especially ade-

quate to some answers formats, and we briefly discuss them in the following section.

2.4 Answer

Anecdotal evidence and at least one study (Barros, 2018), points to the students' preference for **computer-based** programming exams vs. **paper-based** ones. The main reason seems to be the reassurance given by the possibility to compile, run and test the code. Even more important is the possibility to automatically test computer code freeing the teacher to evaluate the non-functional aspects, such as code style, simplicity, and documentation. Also, it seems clear that programming competencies are better assessed by computer-based exams when compared to paper-based ones (Kalogeropoulos et al., 2013) and also better than programming assignments (Daly and Waldron, 2004).

Multi-choice answers also allow automatic grading and are also helpful as formative assessment by allowing each choice to explain being right or wrong.

Finally, the request for **descriptive** (theoretical) answers is probably the least liked by students and teachers. They are seen as nearly useless to check coding ability and can take much more time and effort to assess and grade.

2.5 Cardinality: Single vs Multiple

It should seem obvious that **multiple** assessment instruments are better than a **single** one. The only rationale for a single assignment (including formative and summative) would be to minimize teacher workload. However, other solutions should be considered even in those cases, e.g., automatic or semi-automatic assessment of programming assignments. Furthermore, a mix of different assessment activities should be perceived as fairer as it becomes easier to assess all learning outcomes better. Nevertheless, an evaluation of teacher and student workload must always be present.

Also, it has been argued that some examinations assess many distinct and heterogeneous concepts resulting in more complex tasks, which should be avoided by the use of decomposition in more atomic elements (Luxton-Reilly et al., 2017). Using multiple activities to be assessed independently would be a way to comply with that concern.

Finally, when several assessment activities are used, one must decide on the relative weights and types. The decisions for each activity assessment/grading should also be considered for the set of activities, as each activity can be seen as a part of one unique large activity. In this sense, some activities can be **mandatory (minimal grade)**, an **weighted average** can be used, some can **improve previous** ones, or the grader can opt to **remove some**, e.g., the highest-graded and the lowest-graded.

2.6 Assessment/grading

Grading can be based on the often wrong assumption that students' abilities are normally distributed. This can motivate the use of a **norm-referenced** grading, usually named "grading on the curve". A **criterion-referenced** grading approach is a much better approach as it will measure if each student achieved the intended learning outcomes. The guide by Nilson presents an excellent presentation of these two approaches to grading (Nilson, 2003). Grading is traditionally made in an **analytic** way: the assessment of the stated requirement is decomposed into several parts, and each part is given a grade. Then each part is assessed in a **all or nothing / binary** form or in a **scale/n-ary** way, using **scale**. This scale can then correspond to a **quantitative** or **qualitative** grade. Although it is usually argued that an analytic approach is more "objective," it only splits the subjectivity into smaller additive pieces. In a **human-based assessment** (be it a **teacher assessment** or a **peer assessment**), each of these pieces will have to be accessed subjectively. Differently, an **holistic** assessment looks at the student answer in a global way (Thompson, 2007). It is especially adequate for a small number of students when it is possible for the teacher to mentally order each answer by its perceived (and subjective!) quality. Ideally, this relative ordering should be complemented by feedback to students. Interestingly, and maybe unsurprisingly, the holistic approach can be as effective and fair as the analytic one. One study found a strong correlation between the results of the two approaches, and the grader's accumulated experience probably plays a significant role in this result (Fitzgerald et al., 2013).

Rubrics can help the grader or graders maintain consistency across many student submissions and can be seen used as a support for a holistic approach to assessment. However, multiple graders are always a risk, so careful attention to each rubric interpretation is needed to guarantee coherence in the applied criteria (Cigas et al., 2018).

In every single assessment, some parts can be **mandatory** in the sense that their completion is needed for approval, or **optional**. The latter can be counted **for normal credit** or **for extra credit** (as a bonus). The extra credit is especially interesting to promote initiative and creativity in larger home assignments. But, it can also be a way to reward more

sophisticated or complete answers.

The extra credit contributes to the **additive** part of grading. Although an additive weighted average is probably the most used form for grading calculation, it is also interesting to apply a **subtractive** part. This allows a set of penalties for demanded "minimal" requirements for which there is no acceptable reason not to fulfill them, e.g., style rules, submission date, or the use of some specific library. The previously mentioned bonus provides a complement for these additive and subtractive parts.

A crucial aspect in large classes is the possibility of **machine-based assessment** by automatic or semi-automatic grading. Nevertheless, the tools, usually test-driven based or inspired by contest tools, often do not give feedback and are challenging to adapt to specific needs (Keuning et al., 2016; Ahoniemi and Reinikainen, 2006).

Besides the traditional **teacher assessment** (either for the **individual assessment** of **group assessment** of students) the possibility of **peer assessment** should be considered (e.g, (Indriasari et al., 2020)).

Finally, although a **n-ary** scale is by far the more commonly used, a **binary** scale, where each task is graded as passed or failed, can provide a good basis for repeated submissions of the same assignment, thus fostering deeper learning. This is especially true if adequate feedback is given for each submission.

3 CONCLUSIONS

Assessment choices are plenty, and programming courses open even more possibilities due to the possibility of practical machine support. The presented guide provides a basis for reflection and is easily adapted to specific course needs and teachers' preferences. A vast body of knowledge on teaching and learning computer programming should be taken into account when designing courses. This paper and the presented guide are contributions in that direction.

REFERENCES

Ahoniemi, T. and Reinikainen, T. (2006). Aloha - a grading tool for semi-automatic assessment of mass programming courses. In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Baltic Sea '06, page 139–140, New York, NY, USA. Association for Computing Machinery.

Barros, J. P. (2018). Students' perceptions of paper-based vs. computer-based testing in an introductory programming course. In *CSEDU 2018-Proceedings of the 10th International Conference on Computer Supported Education*, volume 2, pages 303–308. SciTePress.

Bennedsen, J. and Caspersen, M. E. (2007). Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32–36.

Bennedsen, J. and Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2):30–36.

Biggs, J. and Tang, C. (2011). *Teaching for Quality Learning at University*. Open University Press, 4 edition.

Brown, S., Race, P., and Smith, B. (2004). *500 Tips on Assessment*. Routledge, 2 edition.

Cigas, J., Decker, A., Furman, C., and Gallagher, T. (2018). How am i going to grade all these assignments? thinking about rubrics in the large. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 543–544, New York, NY, USA. Association for Computing Machinery.

Daly, C. and Waldron, J. (2004). Assessing the assessment of programming ability. *SIGCSE Bull.*, 36(1):210–213.

de Raadt, M. (2012). Student created cheat-sheets in examinations: Impact on student outcomes. In *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123*, ACE '12, page 71–76, AUS. Australian Computer Society, Inc.

Fitzgerald, S., Hanks, B., Lister, R., McCauley, R., and Murphy, L. (2013). What are we thinking when we grade programs? In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 471–476, New York, NY, USA. Association for Computing Machinery.

Garg, M. and Goel, A. (2022). A systematic literature review on online assessment security: Current challenges and integrity strategies. *Computers & Security*, 113:102544.

Hwang, C. J. and Gibson, D. E. (1982). Using an effective grading method for preventing plagiarism of programming assignments. *SIGCSE Bull.*, 14(1):50–59.

Indriasari, T. D., Luxton-Reilly, A., and Denny, P. (2020). A review of peer code review in higher education. *ACM Trans. Comput. Educ.*, 20(3).

Kalogeropoulos, N., Tzigounakis, I., Pavlatou, E. A., and Boudouvis, A. G. (2013). Computer-based assessment of student performance in programing courses. *Computer Applications in Engineering Education*, 21(4):671–683.

Keuning, H., Jeuring, J., and Heeren, B. (2016). Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, page 41–46, New York, NY, USA. Association for Computing Machinery.

Luxton-Reilly, A., Becker, B. A., Cao, Y., McDermott, R., Mirolo, C., Mühling, A. M., Petersen, A., Sanders, K., Simon, and Whalley, J. L. (2017). Developing assessments to determine mastery of programming funda-

mentals. *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*.

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018). Introductory programming: A systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 2018 Companion, page 55–106, New York, NY, USA. Association for Computing Machinery.

Malmi, L. and Korhonen, A. (2008). Active learning and examination methods in a data structures and algorithms course. In Bennedsen, J., Caspersen, M. E., and Kölling, M., editors, *Reflections on the Teaching of Programming: Methods and Implementations*, pages 210–227. Springer Berlin Heidelberg, Berlin, Heidelberg.

Medeiros, R. P., Ramalho, G. L., and Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90.

Nilson, L. B. (2003). *Teaching at Its Best A Research-BAsed Resource for College Instructors*. Anker Publishing Company, Inc., 2 edition.

Nørmark, K., Thomsen, L. L., and Torp, K. (2008). Mini project programming exams. In Bennedsen, J., Caspersen, M. E., and Kölling, M., editors, *Reflections on the Teaching of Programming: Methods and Implementations*, pages 228–242. Springer Berlin Heidelberg, Berlin, Heidelberg.

Quille, K., Nolan, K., Becker, B. A., and McHugh, S. (2021). *Developing an Open-Book Online Exam for Final Year Students*, page 338–344. Association for Computing Machinery, New York, NY, USA.

Salguero, A., Griswold, W. G., Alvarado, C., and Porter, L. (2021). Understanding sources of student struggle in early computer science courses. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, ICER 2021, page 319–333, New York, NY, USA. Association for Computing Machinery.

Salguero, A., McAuley, J., Simon, B., and Porter, L. (2020). A longitudinal evaluation of a best practices cs1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ICER '20, page 182–193, New York, NY, USA. Association for Computing Machinery.

Stuurman, S., Passier, H. J., Geven, F., and Barendsen, E. (2019). Autism: Implications for inclusive education with respect to software engineering. In *Proceedings of the 8th Computer Science Education Research Conference*, CSERC '19, page 15–25, New York, NY, USA. Association for Computing Machinery.

Thompson, E. (2007). Holistic assessment criteria: Applying solo to programming projects. In *Proceedings of the Ninth Australasian Conference on Computing Education - Volume 66*, ACE '07, page 155–162, AUS. Australian Computer Society, Inc.

Williams, L., Titus, K. J., and Pittman, J. M. (2021). How early is early enough: Correlating student performance with final grades. In *Computing Education Practice 2021*, CEP '21, page 13–16, New York, NY, USA. Association for Computing Machinery.