Informatics in Education, 2023, Vol. 22, No. 2, 209–232

DOI: 10.15388/infedu.2023.09

On the Use of Feedback in Learning Computer Programming by Novices: A Systematic Literature Mapping

Hemilis Joyse BARBOSA ROCHA[1],
Patrícia CABRAL DE AZEVEDO RESTELLI TEDESCO[1],
Evandro DE BARROS COSTA[2]

[1]*Informatics Center, Federal University of Pernambuco, Brazil*
[2]*Computing Institute, Federal University of Alagoas, Brazil*
e-mail: hjbr@cin.ufpe.br, pcart@cin.ufpe.br, evandro@ic.ufal.br

Received: September 2021

Abstract. In programming problem solving activities, sometimes, students need feedback to progress in the course, being positively affected by the received feedback. This paper presents an overview of the state of the art and practice of the feedback approaches on introductory programming. To this end, we have carried out a systematic literature mapping to understand and discuss the main approaches for providing and evaluating feedback used in the learning of novice programmers in the problem-solving activity. Thus, according to a formal protocol, an automatic search was performed for papers from 2016 to 2021. As a result, 39 studies were selected for the final analysis. As a result, we propose three different categorizations: the main approaches to providing feedback, the main methods used in the evaluation and the main aspects and effects of the evaluated feedback.

Keywords: feedback, computer programming, novices.

1. Introduction

The Problem solving skill is one of the eight key competences for sustainability in the 21st century defined by UNESCO (2017). This skill has been educationally explored in the most diverse areas of knowledge, such as mathematics and computer programming. In the latter, Medeiros *et al.* (2018), Kunkle and Allen (2016) and Koulouri *et al.* (2014) consider that problem solving is an inherent skill in the computer programming learning process. In particular, aligned with these authors, Mathew *et al.* (2019) highlights the relevance of the problem solving skill for introductory programming (IP) courses for beginners.

Studies on computer education reveal that learning to program, therefore requiring students' problem solving skills, is a difficult task, with high failure rates (Kunkle and Allen, 2016; Koulouri *et al.*, 2014). This situation of academic failure of students in relation to learning programming has led to the development of research aimed at alleviating this type of problem or identifying its causes. In this sense, Veerasamy *et al.* (2019) conducted a study and concluded that students with the profile of problem solvers presented better grades in the programming disciplines. There are also some studies that propose solutions to support programming learning with computational problem solving (Chao, 2016; Loksa *et al.*, 2016). Despite this, in the problem solving process, students sometimes need help to progress through the (Caspersen and Bennedsen, 2007) course. Thus, this help can happen through feedback, where the instructor has the opportunity to guide the students in order to improve their performance (Langer, 2011; Brusilovsky and Weber, 1996).

Hattie and Timperley (2007), Shute (2008) and Nicol and Macfarlane-Dick (2006) emphasize that the provision of quality feedback is crucial to the level of student success. In the computer programming domain, effective feedback is also considered an essential element in the student learning process(Corbett and Anderson, 2001; Gusukuma *et al.*, 2018; Marwan *et al.*, 2019; Becker *et al.*, 2018). While the authors agree on the importance of good feedback, they also report that it requires a large investment of time by teachers Nguyen *et al.* (2014). Aiming to reduce the instructor's effort and to improve feedback quality, many researchers have proposed computational tools that help (Perera *et al.*, 2021) programmers' teaching and learning.

On the other hand, although feedback plays a central role in student development, according to Hattie and Timperley (2007), poor feedback can cause problems in student learning and even lead to dropout. Therefore, to classify feedback as good or bad, an evaluation is necessary. (Smith *et al.*, 2017; Stephens-Martinez and Fox, 2018; Cavalcanti *et al.*, 2020).

Considering that novice programmers are very much affected by the feedback according to (Marwan *et al.*, 2020), we want to know how feedback has been provided to this audience. To this purpose, we carried out a systematic literature mapping (SLM), seeking answers to the following research questions:

RQ01: What are the main approaches used to generate feedback during learning problem solving in the computer programming domain for beginners?

RQ02: How has the feedback provided in the learning problem solving activity in the computer programming domain for beginners been evaluated?

Thus, to address these two research questions, we present an overview of the current state of the art of research carried out in the period from 2016 to 2021. More specifically, we characterize the approaches and forms of evaluating the feedback provided to secondary and technical students, as well as to undergraduate students. Thus, the results obtained are discussed, pointing out, above all, how the feedback has been provided, how it has been evaluated. Furthermore, where there are more gaps and open questions, therefore guiding potential relevant research.

This article is organized as follows. In section 2, we present the background of concepts related to the research topic. In section 3, we describe the systematic method ad-

opted and in section 4, we detail the definition and execution of the systematic literature mapping. In section 5, we present and analyze the data of the results found. In section 6, we present the discussions of the results found. Final considerations and future work are described in Section 7.

2. Background

This section presents some aspects of feedback, focusing on programming learning.

2.1. *Overview*

Glassey (2019) says that the feedback process is a cycle that includes providing feedback, receiving it and implementing it in the task. For Cano (2013) feedback is the process by which students obtain information about their work, evaluating the similarities and differences between the appropriate standards for any work, and the qualities of the work itself, in order to generate an improved work.

The important role of feedback in guiding the learning process and supporting the improvement of progress in student performance is widely recognized in the (Belcadhi, 2016) literature. In this sense, feedback can be used by the teacher as a resource to identify students' needs and thus adapt their methods and contents (Orrell, 2006). The use of this resource can help to mitigate situations where, many times, low students' performance can be associated with poor understanding of the requirements of the tasks proposed to them (Rust *et al.*, 2003).

Through feedback, the teacher has the opportunity to guide the students in order to improve their performance, self-efficacy and self-regulation. Friedman (2015) was the first to conceptualize self-efficacy and defined it as one's beliefs in their ability to organize and execute courses of action required to produce certain achievements. The work in Zimmerman (2013) characterizes self-regulation as the students' own control and regulation of their thoughts, cognition, affection, motivation, behavior and environment, in favor of academic goals that are feedback by means of the student's own learning experiences. Thus, for the student, feedback can help identify areas of improvement in their knowledge or skills, and reflect on their learning strategies (Parikh *et al.*, 2001). Hence, we can understand feedback as a learning thermometer for the students and through it it is possible to identify their strengths and weaknesses.

2.2. *Quality Aspects in Feedback*

Boud and Falchikov (2007) emphasizes that bad feedback can be detrimental to students' self-efficacy and motivation and, therefore, creating distrust in the feedback process and in the teacher. This finding, according to these authors, has motivated the development of research in various perspectives of feedback reporting the benefits and impact on learning.

According to Carless *et al.* (2011) a study carried out in Australia and the United Kingdom found that students are often dissatisfied with the feedback provided. More specifically, they found that the accuracy, timeliness and consistency of the feedback information were lacking. Jessop *et al.* (2014) reports that there are wide variations in feedback practices, as well as inconsistencies in the quality and quantity of feedback. On the other hand, providing effective feedback to students is considered a key resource in the learning process. Effective feedback is understood as appropriate and timely feedback (Mory, 2004). Being opportune, when it is provided it suits the needs of the situation (Knight and Yorke, 2003), and appropriate if it is sufficient (Holmes and Smith, 2003), that is, the information given in the feedback message is sufficient for the student or a more detailed and complete feedback is needed.

Nicol and Macfarlane-Dick (2006) carried out a study where they defined seven principles of good feedback practices. For these authors, they are considered good feedback practices, when the feedback:

(1) Helps to clarify what good performance is.
(2) Facilitates self-assessment.
(3) Provides high-quality information to students about their learning.
(4) Encourages dialogue about learning.
(5) Encourages positive motivational beliefs and self-esteem.
(6) Offers ways to bridge gaps between the current state and the learning objective.
(7) Helps teachers shape teaching.

Principles of good feedback practices can be found widely in the literature. Thus, researchers have been developing work, some focusing on understanding the student's perception and how they should react to receiving feedback.

2.2.1. *Focus on the Student*

Sadler (1989) has developed a theory of formative assessment where, in order to take better advantage of the feedback provided, the student needs to have a goal, compare current performance with the goal, and take actions to bridge the gap between the current state and the learning goal. In this sense, Mutch (2003) and Weaver (2006) try to identify how students should receive and act when getting feedback, analyzing, for example, its readability. In that respect, Poulos and Mahony (2008) carried out a study to identify the perceptions, impact and credibility of the feedback provided from the student's perspective.

Some authors suggest that students should acquire a set of learning strategies that provide them with a foundation for taking responsibility and personal control over their learning process (Schunk and Zimmerman, 1998; Zimmerman, 2013). Thus, applying the idea of self-regulation introduced by Schunk and Zimmerman (1998), who state that for a good feedback to be achieved, the instructor must strengthen the student's self-regulation capacity Nicol and Macfarlane-Dick (2006).

Butler and Winne (1995) developed a model of self-regulation of learning. It explores the student's ability to generate internal feedback at all stages of the learning process.

2.3. *Feedback on Learning Programming*

The first experience of learning programming for many students is often frustrating. Sheard *et al.* (2009) are some of the authors who observed the volume of works that point to the difficulties of introductory programming as a subject considered difficult to learn and teach. Lahtinen *et al.* (2005) state that the biggest problem experienced by beginners in programming does not seem to be the understanding of the basic concepts of programming logic, but the combination and proper use of these concepts in the construction of a given program. That is, "putting the program pieces together" (Spohrer and Soloway, 1989).

In the perspective of tackling the problems mentioned above, the literature emphasizes the importance of feedback, especially to support students in the activity of solving programming problems. Hattie and Timperley (2007) identified four levels of feedback: task level, process level, self-regulation level and self-level (Ott *et al.*, 2016) use these levels to compose a feedback model for novice programmers. In this model, they describe feedback for programming at the task, process, and self-regulation levels.

Aiming to improve the quality of feedback, there are some proposals for computational tools to assist in the teaching and learning of (Perera *et al.*, 2021) programming. Some tools with feedback for the solution submitted by the student (Kumar, 2006) and others that provide tips for the student to find the solution (Al-Imamy *et al.*, 2006).

3. Related Work

When analyzing the literature, we notice that there are several works with review or mapping studies in the teaching and learning of computer programming domain, considering different aspects. These studies analyze topics such as: aspects and evidence on introductory programming (Luxton-Reilly *et al.*, 2018); misconceptions, student misconceptions in introductory programming (Qian and Lehman, 2017); elements of programming languages and educational platforms (Perera *et al.*, 2021); and previous skills, basic knowledge of (Medeiros *et al.*, 2018) students. However, the works by Keuning *et al.* (2016) and Keuning *et al.* (2018), present research specifically related to feedback in the programming domain, and thus they were selected as the most strongly related to this systematic mapping.

Keuning *et al.* (2016) published the results obtained, using the snowballing technique, intended to discover, in the teaching tools that provided feedback, its type, the techniques used to generate it, its adaptability and how these tools are evaluated. Later, including a database search, through the review with publications until 2015, they complemented the research results by publishing in Keuning *et al.* (2018). Overall, we consider that the studies discussed above make a relevant contribution to the area. However, such studies do not focus on a specific target audience for programming students.

In this study, we focused on novice programmers as our target audience and the motivation for this choice was found in the work of Marwan *et al.* (2020). In this work, the author states that novice programmers are very affected by the feedback provided. Thus, in this article, we present a systematic mapping seeking to characterize the provision of feedback for beginning programmers in problem solving.

4. Definition and Execution of the Systematic Literature Mapping

The SLM was conducted in accordance with the guidelines proposed by Kitchenham and Charters (2007). Thus, we will present the implementation of the SLM in two subsections: definition of the review protocol and the search process.

4.1. *Definition of the Review Protocol*

Considering that the definition of the protocol has a significant impact on the quality of the SLM, it is necessary to validate the (Kitchenham and Charters, 2007) protocol. This validation can be performed through a pilot test, whose objective is to verify the feasibility of carrying out the review, allowing the identification of necessary adjustments.

The protocol definition was carried out in three steps, as shown in Fig. 1:

i) **Pilot protocol definition:** we formulated the protocol for the pilot test.
ii) **Pilot protocol execution:** we applied the protocol and retrieved 40 studies for the control group.
iii) **Definition of the final protocol:** we analyzed the studies from the control group and made adjustments to the words and connectives of the string search, added exclusion criteria, refined the inclusion criteria and enriched the research questions.

The final version of the SLM protocol with all its elements: research questions, exclusion criteria, inclusion criteria and search *string* is presented below.

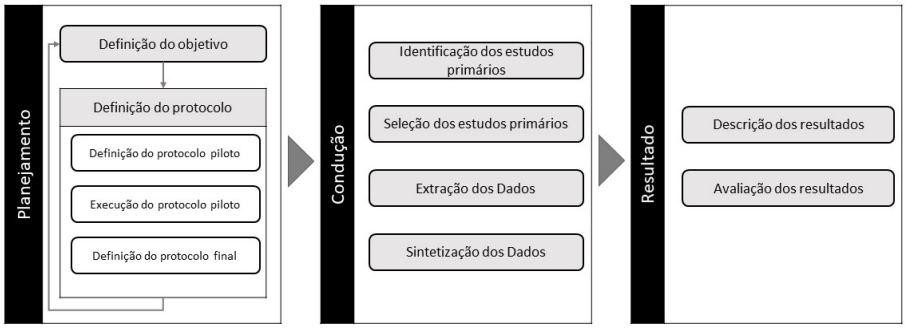

Fig. 1. Steps in the review protocol.

Table 1
PICO

Population	Current studies on the provision of feedback in problem solving activity in learning programming for beginners.
Intervention	Assessment aspects and approaches to providing feedback in the problem solving activity.
Comparison	Not applicable.
Outcomes	Methodologies, computational systems and methodological tools for providing feedback in problem solving activities.

Table 2
Questions

	Questions	Motivation
RQ01:	What are the main approaches used to generate feedback during learning problem solving in the computer programming domain for beginners?	As an answer to this question, we seek to catalog the studies that present some methodology or computational system for providing feedback in the problem solving activity, in the domain of learning computer programming for beginners.
RQ02:	How has the feedback provided in the learning problem solving activity in the computer programming domain for beginners been evaluated?	By answering this question, we sought to characterize which methodological instruments are being used and the aspects of feedback are being considered in the evaluations of the feedback provided in the problem solving activity, in the domain of teaching and learning computer programming for beginners.

4.1.1. *Research Questions*

A commonly used approach to formulating research questions is to use the PICO criteria. Using PICO, the research questions are structured in four aspects: population, intervention, comparison and Kitchenham and Charters (2007) results. Table 1 shows the PICO attributes defined for the mapping performed.

To identify approaches to problem solving activity in teaching and learning computer programming for beginners, two research questions were considered. They are presented together with their motivations in Table 2.

4.1.2. *Inclusion and Exclusion Criteria*

After defining the research questions, in order to select the studies, the inclusion and exclusion criteria were elaborated. Thus, a study must be excluded if it meets any of the exclusion criteria presented in Table 3.

In addition, for a study to be included in the SLM, it has to satisfy the inclusion criteria (IC) listed in Table 4.

Based on the Kitchenham and Charters (2007) guidelines, we developed two quality criteria: (1) clarity and consistency in reporting on the links between data, interpretation, and conclusions; and (2) clarity in describing the approach to feedback.

Table 3
Exclusion criteria

	Exclusion criteria
CE01:	The studies don't discuss the provision of feedback to solve problems in teaching and learning computer programming for novices
CE02:	The paper is not written in English
CE03:	The paper is not available for download
CE04:	The file is a technical report, lecture notes, slide, poster, position paper, secondary study, editorial, tutorial or short paper
CE05:	The paper is a copy or an older version of another article already considered

Table 4
Inclusion criteria

	Inclusion criteria
CI01:	The study deals with providing feedback for solving problems in teaching and learning computer programming for novices
CI02:	The study discusses the teaching of programming aimed at secondary, technical or higher education
CI03:	The study is available in English in a full paper category
CI04:	The article was published from 2016 to 2021

Table 5
String

String
feedback AND (student OR beginner OR novice) AND (learning OR teaching) AND programming

4.1.3. *Search String*

After carrying out the process shown in Fig. 1, we have prepared the search String presented in Table 5.

4.2. *Search, Selection and Classification of Studies*

After defining and validating the protocol defined in the previous section, we started the application of the protocol starting with the search for studies and followed with the selection and classification of studies.

4.3. *Search Process*

To carry out the study search process, we adopted the following databases for the SLM: ACM Digital Library (`www.dl.acm.org`), IEEE Xplore (`www.ieee.org`) and Science Direct (`www.sciencedirect.com`). In Table 6, we present the search String together with the filters, applied in the configuration of each engine, used in each source. These bases were chosen because they are considered the main ones for the area.

4.3.1. *Selection and Classification*

After recovering the studies, we performed the selection and classification of the studies. Thus, the primary studies were selected and classified according to the exclusion criteria in two stages.

In the first stage, the pre-selection of studies was conducted in two moments. At first, the review of the article characteristics was carried out by two reviewers. Thus, the title, abstract and keywords of the studies retrieved from each source and each selected study according to the exclusion criteria were analyzed. In the second moment, during meetings, the reviewers discussed possible disagreements about the selection or not of a study. We tried to eliminate as many irrelevant studies as possible, taking care that no relevant studies were discarded. As a result, a total of 169 studies were selected.

In the second stage, the full texts of the pre-selected primary studies from the first stage were obtained. The full text of each selected primary study was read by the reviewers, applying the inclusion and quality criteria. And finally, the primary studies included in the final selection correspond to the relevant articles that answered the research questions addressed in this SLM. Thus, 39 relevant studies were identified.

Since in most studies that address feedback for teaching computer programming, the target audience and the type of programming activity are not very clear, we decided to create a String that would encompass more articles. That is, there are studies discussed in this mapping that, despite the target audience being novice programmers and provid-

Table 6
Source

Source	Query	Filter	Number of studies
ACM Digital Library	(“feedback” AND (“student” OR “beginner” OR “novice”) AND (“learning” OR “teaching”) AND “programming”)	Article Type: Research Article Search Within: Fulltext Publication Date: 2015 to 2021	3,820
IEEE Xplore	feedback AND (student OR beginner OR novice) AND (learning OR teaching) AND programming	Search Within: Full text Publication Date: 2015 to 2021	282
Science Direct	(“feedback” AND (“student” OR “beginner” OR “novice”) AND (“learning” OR “teaching”) AND “programming”)	Article Type: Research Article Publication title: Computers E Education Publication Date: 2015 to 2021	657
		Total	4,759

ing feedback in the problem solving process, they were not being captured with other configurations of our Search String. Therefore, we decided that the initial 5000 works would be analyzed to avoid any important study being left out and, during the analysis, we realized that there was a lot of duplication in the retrieved studies. In addition, the application of inclusion and exclusion criteria CE01 and CI01 was responsible for selecting 95% of the works.

5. Data Analysis

Through the elaboration of graphics, tables and discussions, we will present in this section, the information extracted from the 39 selected articles. First, in Section 5.1, we describe the survey we seek to get an overview of all recovered surveys. Then, in Section 5.2, we analyze the issues in Section 4.1.1. In Table 7 we list all selected works.

Table 7
Selected papers

ID	References
AL21	(Alwabel, 2021)
AH18	(Ahmed *et al.*, 2018)
AH19	(Ahmed *et al.*, 2019)
AH20	(Ahmed *et al.*, 2020)
BE18	(Benotti *et al.*, 2018)
BH18	(Bhatia *et al.*, 2018)
CH17	(Chow *et al.*, 2017)
CO16	(Combéfis and Schils, 2016)
FE19	(Feldman *et al.*, 2019)
DA19	(Day *et al.*, 2019) (Day *et al.*, 2019)
DE21	(Denny *et al.*, 2021) (Denny *et al.*, 2021)
ED20	(Edmison and Edwards, 2020)
ED17	(Edwards and Murali, 2017)
EN19	(Endres *et al.*, 2019)
FA18	(Fabic *et al.*, 2018)
GA16	(Gao *et al.*, 2016)
HA19	(Hajja *et al.*, 2019)
HA18	(Haldeman *et al.*, 2018)
HR19	(Hameer and Pientka, 2019)
HO19	(Höppner, 2019)
IN21	(Indriasari *et al.*, 2021)
JE20	(Jemmali *et al.*, 2020)
JI20	(Jiang *et al.*, 2020)
KA18	(Kadekar *et al.*, 2018)
KY16	(Kyrilov and Noelle, 2016)
LA17	(Latih *et al.*, 2017)
LO16	(Lobb and Harlow, 2016)

Continued on next page

Table 7 – continued from previous page

ID	References
MA20	(Marwan *et al.*, 2020)
ME16	(Mendoza *et al.*, 2016)
RE20	(Renzella and Cain, 2020)
SM17	(Smith *et al.*, 2017)
SM19	(Smith *et al.*, 2019)
ST16	(Staubitz *et al.*, 2016)
TR18	(Treviño and Cavazos, 2018)
WA18	(Wang *et al.*, 2018)
WA20	(Wang *et al.*, 2020)
YA20	(Yan *et al.*, 2020)
LI19	(Liu *et al.*, 2019)
GU18	(Gusukuma *et al.*, 2018)

5.1. *General Analysis*

The selected works are geographically distributed through 16 countries: Taiwan, Saudi Arabia, Ireland, Sweden, UK, New Zealand, Mexico, Malaysia, India, USA, Canada, Belgium, Australia, Argentina, Germany and Singapore. Thus, we can observe that the USA has a total of 26 studies, 52%, concentrating the vast majority of the selected research initiatives.

In Section 4.1.2, among the inclusion criteria, we elaborated one that specifies the levels of education of novice programmers, we are interested in. For this reason, we show in Fig. 2 the percentage of selected studies for each level of education. We can observe that there is a predominance of 84% of studies aimed at novice undergraduate programmers, 7% for high school, 7% for technical education and a small percentage of 2% of studies that were developed for any the level of education of the novice programmer.

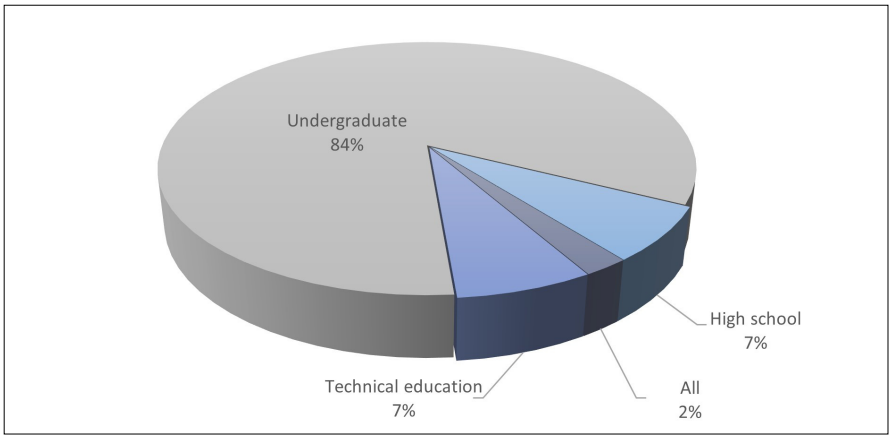

Fig. 2. Distribution of surveys by level of education

5.2. *Analysis Question*

To answer the research questions, we extracted relevant information from the encountered articles as specified in Tables 8, 9 and 10. The articles were read and the data extracted by the authors of this paper.

5.2.1. ***RQ01:*** *What Are the Main Approaches Used to Generate Feedback During Learning Problem Solving in the Computer Programming Domain for Beginners?*

In the search for answers to this question, we performed an analysis of 39 articles and identified some explicit or implicit approach to generating feedback in 29 studies. To identify the approaches in the studies, we created a spreadsheet where the following information was extracted for each study: year, title, authors, identification of the approach and description of the approach. As a result of the process, we extracted six types of analyzed works and we explain each one of them below:

- **Feedback with test cases (WTS)**: in this category, feedback is generated based on test case. Thus, the feedback message can contain one or more of the following information: table showing the tests that were applied in the student's solution, input suggestions for students to test their code, or quality of test cases.
- **Feedback with counterexample (WAE)**: some authors follow in their studies that providing feedback using counterexamples can also be beneficial to the student. Thus, the feedback in this category is characterized by the association of a counterexample to each error made by the student.
- **Feedback with corrections for solution (CFS)**: in this category, feedback is generated in the form of corrections either for the student's entire solution or part of it.
- **Feedback with example of similar solution (ESS)**: in this approach, the generated feedback contains a suggestion of a correct code example related to the student error characteristics.
- **Feedback guide to the solution (GTS)**: for generating feedback, the solution that the student is building is considered, through tips, guiding them towards the correct solution.
- **Feedback with error and hits identification (EHI)**: the feedback message contains identifications of possible errors or successes of the student,with the identification of the lines of code with errors.

In Table 8 we present each work associated with the respective approaches.

In Fig. 3 we display the percentage of articles classified in each category. In this figure, we can see that 43 % of the studies work on feedback with identification of errors and successes, followed by 22% of feedback with corrections for the solution, 11% of feedback with test cases and feedback guide for the solution, feedback with an example of similar solution and feedback with counterexamples, all with the same percentage of 8%.

It was also possible to observe that studies sometimes present more than one approach to feedback. Of the 16 works involving the EHI approach, 1 also provides ESS,

Table 8
Information extracted from analysis question 1

Approaches	References
Feedback with test cases	LO16, CH17, SM17 and EN19
Feedback with counterexample	GA16 and LI19
Feedback with corrections for solution	DA19, AH18, HR19, JI20, RE20, WA20, HA19 and AH19
Feedback with example of similar solution	WA20, AH19 and JI20
Feedback guide to the solution	SM19, FE19 and HA18
Feedback with error and hits identification	AL21, BE18, CO16, DA19, ED17, EN19, FE19, MA20, ME16, RE20, FA18, MA20, FA18, ST16, BH18, WA18, HO19 and JE20

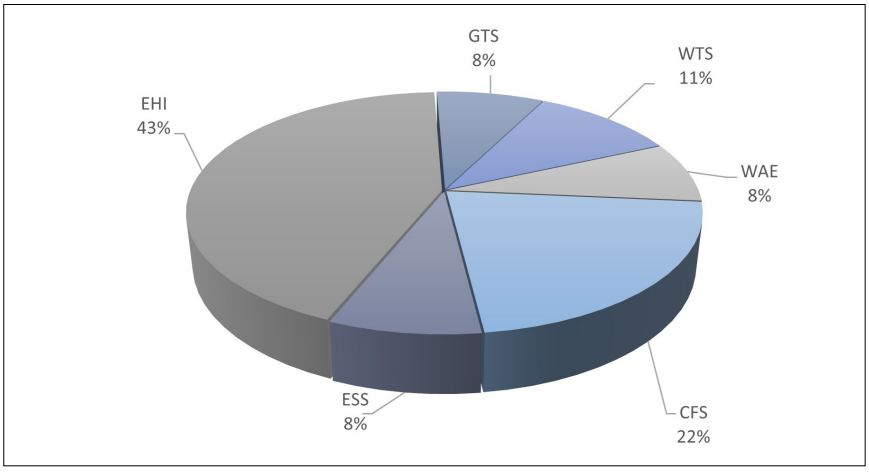

Fig. 3. Percentage of works for each category of approach

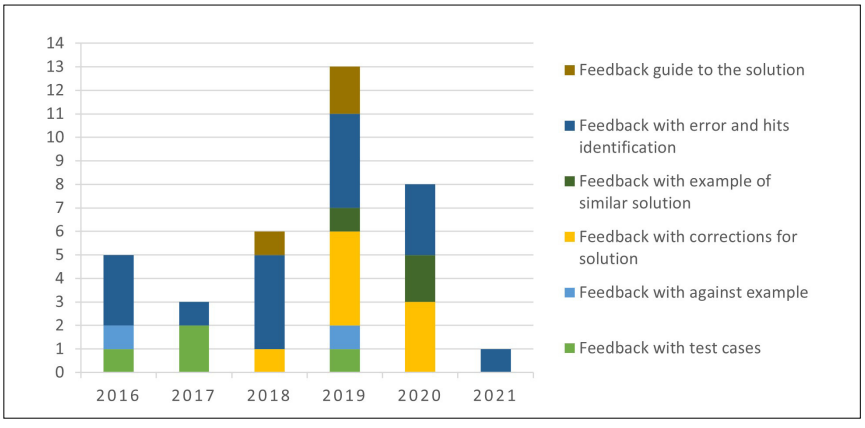

Fig. 4. Temporal distribution of categories of approaches.

which in turn, of the 3 studies with a feedback approach with ESS, 1 also implements the CFS approach.

Analyzing the graph in Fig. 4, we can see that 2019 is the year with the largest number of published studies covering all the categories of approaches discussed above. On the other hand, in 2021 there is only one work with one approach.

5.2.2. ***RQ02:*** *How Has the Feedback Provided in the Learning Problem Solving Activity in the Computer Programming Domain for Beginners Been Evaluated?*

To answer RQ02, we invested in a deeper analysis of the studies, systematizing the results in an electronic spreadsheet. Thus, we identified 15 works, out of a total of 39 selected in the systematic mapping, and with that, we will point out the main aspects of evaluation of the feedback provided to novice programmers. To that end, we carried out a two-step process: identification of evaluation methods, evaluated aspects and effects and grouping of evaluated methods, aspects and effects. For the phase of identification of the evaluation methods, evaluated aspects and effects, we created a spreadsheet where, for each study, the following information was extracted: year, title, authors, method identification, method description, aspect identifier, aspect description, effect identifier, effect description. In the next step, in the method, aspects and effects grouping phase, we concatenate with the same identifier all the methods, aspects or effects that presented the same descriptions. As a result of the process, we extracted four evaluation methods, nine aspects and four effects evaluated in the analyzed works.

Below we categorize the main instruments used in the selected studies to assess the aspects present in Table 10:

- **Experiment data analysis (AD):** This method was applied to assess the effects of feedback on a group of students, through the analysis of data produced by conducting a controlled experiment to analyze the aspects and effects mentioned in Table 10.
- **Interview (IW):** with this method, the feedback provided is evaluated in an interview with the students to find out the students' opinion about the aspects and effects mentioned in the Table 10.
- **Open questionnaire (OQ):** another method used to assess the feedback provided is the application of an open questionnaire to find out the students' opinion about the aspects and effects mentioned in Table 10.

Table 9
Information extracted from research question

Methods	References
Experiment data analysis	KY16, KA18, ED20, AH20, MA20 and TR18
Interview	MA20
Open questionnaire	DA19, RE20 and IN21
Questionnaire using Likert scale	ED17, LA17, TR18, EN19, YA20 and DE21

- **Questionnaire using Likert scale (LS):** the application of this method consistsofusing a Likert scale in a questionnaire after providing feedback to analyze the aspects and effects mentioned in the Table 10.

In the table 9 we present the works allocated to each of the categories of evaluation methods described above.

To better visualize how the works that showed feedback evaluation for novice programmers were distributed over the last six years, we have elaborated Fig. 5. There, we can observe that most of the works, a total of 6, carry out the evaluation of the feedback provided. Most were published in 2020 and and 1 was published in 2016. In addition, we also found that most of the studies evaluated the feedback using the experimental method and data analysis and only one study evaluated it using an interview.

We present below the aspects, effects and the question of investigation of the feedback evaluated, through the methods discussed above, in the selected studies.

- **Identify the error:** Did the feedback help the student find the error?
- **Fix the error:** Did the feedback help the student correct the error?
- **Improve the solution:** Did the feedback help the student to improve the existing solution?
- **Faster completion of the solution:** Did the feedback help the student complete the solution faster?
- **Staying on course:** Did the feedback help the student stay on course?
- **Involvement with the course:** Did the feedback help the student to be more committed to the course?
- **Improve performance in learning programming:** Did the feedback help the student to improve learning performance in the programming subject?
- **Positive effect on grades:** Did the feedback help the student improve their grades?
- **Know the cause of the error:** Did the feedback help the student complete the solution faster?
- **Feedback usefulness:** Did the student find the feedback helpful?
- **Feedback speed:** How quickly was feedback provided?

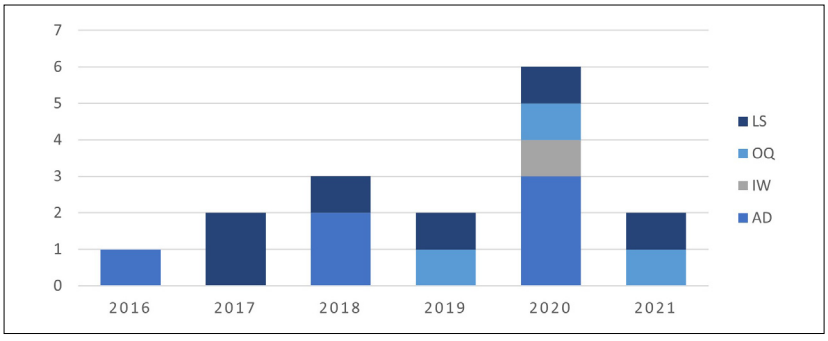

Fig. 5. Temporal distribution of assessment instruments.

- **Completeness of feedback:** How complete does the student consider the feedback?
- **Quality of feedback:** What is the quality of the feedback?

We present in Table 10 the association of each selected work, for this question, to the category of aspects and effects of the evaluated feedback.

We have distributed the works, according to the aspects and feedback effects, in the six years specified for this systematic mapping and presented in Fig. 6. By observing the graph, we notice that the category Faster completion of the solution (FCS) has the largest number of associated works, being published in 2016, 2018, 2020 and 2021. The categories Improve the solution (ITS), Faster completion of the solution (FCS)), Staying on course (SOC), Involvement with the course (ITC), Improve performance in learning programming (PLP), Feedback speed (FES) and Completeness of feedback (COF) only have works published in 2020.

In our last analysis of the selected works on this question, we relate the categories of the methods to the aspects and effects and present the graph in Fig. 7. With this resource,

Table 10
Method extracted: RQ02

	Information extracted	References
Feedback Aspects	Identify the error (ITE)	KY16, KA18 and EN19
	Fix the error (FTE)	KY16, KA18, YA20 and DE21
	Improve the solution (ITS)	ED20 and AH20
	Faster completion of the solution (FCS)	ED20
	Staying on course (SOC)	MA20
	Involvement with the course (ITC)	MA20
	Improve performance in learning programming (PLP)	MA20
	Positive effect on grades (PEG)	TR18
	Know the cause of the error (KCE)	DE21
Feedback Effects	Feedback usefulness (FEU)	DA19, MA20 e LA17
	Feedback speed (FES)	RE20
	Completeness of feedback (COF)	RE20
	Quality of feedback (QOF)	ED17, IN21 and EN19

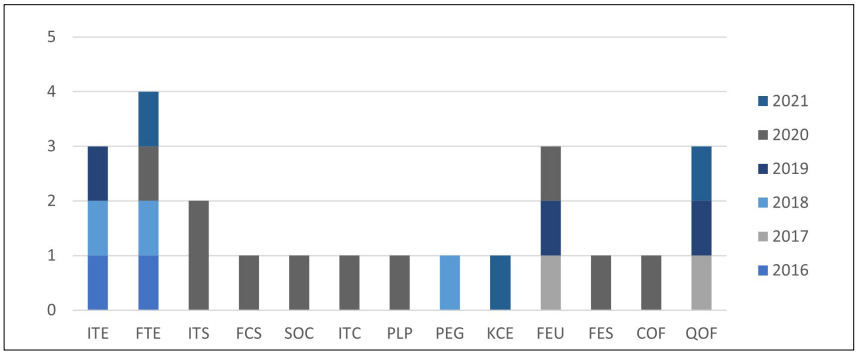

Fig. 6. Time distribution of evaluation aspects.

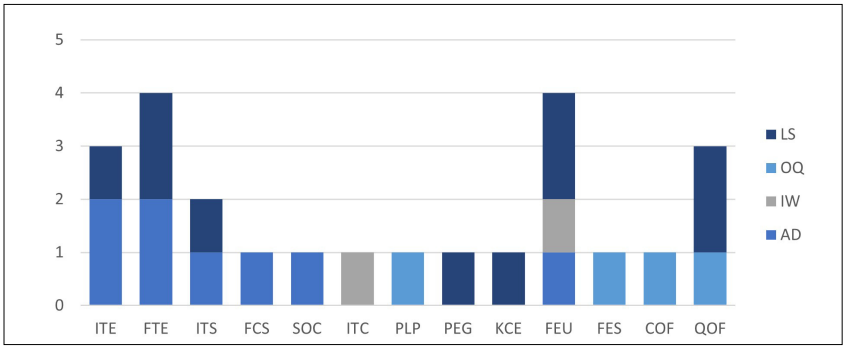

Fig. 7. Relationship assessment instrument versus assessed aspects and effects.

we can make an interpretation that has two dimensions: the categories of aspects and effects evaluated in relation to the categories of evaluation methods and the categories of evaluation methods in relation to the categories of aspects and effects evaluated. In addition, it is noteworthy that the evaluations reported in the works generally consider more than one aspect or effect to be evaluated.

When we analyze the graph from the perspective of the first interpretation dimension, we can see that:

(i) The categories Identify the error (ITE), Improve the solution (ITS) and Fix the error (FTE) aspects were evaluated using the same categories of methods.
(ii) The categories Faster completion of the solution (FCS) and Staying on course (SOC) were evaluated only with the Experiment data analysis method.
(iii) The categories Improve performance in learning programming (PLP), Feedback speed (FES) and Completeness of feedback (COF) were evaluated by only the Open questionnaire method.
(iv) The Feedback usefulness (FEU) category has the greatest variety of evaluation methods.

If we interpret the graph in the sense of the methods in relation to the aspects and effects evaluated, we can carry out the following analyzes: (i) The Questionnaire method using a Likert scale (LS) was used to assess the variety of aspects and effects and it is present in most studies; and (ii) the Interview method (IW) was applied in the smallest variety of aspects and studies.

6. Discussions

As presented in the previous section, the mapping process resulted in the selection of 39 studies. With the extraction of data from the studies, we developed Fig. 2, where we identified a lack of research aimed at providing feedback to students attending secondary and technical courses. For the vast majority of studies have undergraduate students as their target audience, since there are secondary studies whose focus is on these Ott *et al.* (2016) students.

Analyzing the first research question, we realized that the feedback approach with identification of hits and misses is the most present. There was little concern about good feedback practices developed by Nicol and Macfarlane-Dick (2006), as most messages are binary, that is, only informing whether students got it wrong or right. However, in the work of Marwan *et al.* (2020), the authors are concerned with self-regulation, since the tool has an objective detector. Here, the student code is continuously checked in real time to determine what goal the students are working on and whether they have achieved it correctly or not. In addition, it provides motivational messages in case the student fails to reach any goals in the last few minutes of work.

The results obtained with the extraction of data for the second question, point to the evaluation methods, aspects and evaluated effects. Regarding the methods, we grouped the works into 4 methods: Experiment data analysis, Interview, Open questionnaire and Questionnaire using Likert scale. We noticed that the most used methods are Experiment data analysis and Questionnaire using Likert scale. In addition, we found that researchers are more concerned with evaluating whether the feedback provided helps the student in the aspects of error identification, error correction and improvement of the solution. Regarding the effects of feedback, the most evaluated were the usefulness and quality of the feedback.

6.1. *Open Questions*

Some questions for inclusion in the research script about providing feedback in the problem solving process in teaching and learning computer programming emerged from this mapping.

During the analysis of the selected works, we noticed that feedback is not always automatically provided by the system. Therefore, we believe that it is worth an investigation to understand what are the main sources of feedback provided and how feedback for novice programmers has been generated in the systems. Another interesting research question is to understand how the approaches discussed here are presented to the student. You can find out, for example, whether the feedback is presented visually or just text.

The last question whose answer could be extracted from the selected works refers to the solution's diagnostic technique. Thus, we understand that the quality of feedback is also associated with the level of analysis of the student's solution. In this way, it is possible to discover and categorize the main analysis techniques of the solution provided by the student.

On the other hand, for Nguyen *et al.* (2014) helping individual students with their problems requires a large investment of instructor time. Therefore, several works selected in this mapping have solutions for automated, but not personalized, feedback. The personalized feedback approach is the main gap identified in these works. Thus, a possible solution to implement the personalized feedback approach would be through an intelligent tutoring system Pillay (2003). According to Barnes and Stamper (2008) based on historical student data, more advanced automated assessment systems can diagnose student programs and customize feedback for students.

6.2. *Study Limitations*

Like previously published secondary studies, this mapping has the following limitations:

- **Preparation of the search string:** During the definition of the pilot protocol, it was noticed that some authors do not use the words *"novices"* or *"beginner"*, although the study deals with programming for beginners. Therefore, we have also inserted the word *"student"* in the conjunction of the String.
- **Publication bias:** Due to the large number of articles analyzed in the pre-selection stage, important articles may have gone unnoticed in the analysis of titles, abstracts and keywords. To mitigate this limitation, the three authors of this mapping participated in the pre-selection stage of the studies. In addition, the search string was carefully constructed using even a pilot protocol as discussed in Section 4.1.
- **Study data selection and extraction:** Threats related to the data selection and extraction stage have been mitigated by providing a detailed definition of inclusion, exclusion and quality criteria. We defined and documented a protocol for study selection and all authors performed the selection together, discussing the selection until consensus was reached.

7. Final Considerations

In this research, we present the results of a SLM on approaches and forms of feedback assessment for teaching and learning programming for novice programmers. The search was carried out with the goal of selecting studies published internationally in the last 6 years. This search resulted in the pre-selection of 169 articles, among which 39 were included for data extraction.

We performed the extraction of data from the studies using two strategies: overview and view of research questions. For an overview, we categorized studies by educational level and made a geographical distribution of selected studies. As a response to RQ1, we identified six approaches used in providing feedback to novice programmers. As a response to RQ2, we characterized four assessment methods for feedback and 13 assessed aspects and effects.

In future research, which is already underway, we intend to carry out a systematic review addressing the complementary issues raised in Section 6. To that end, the authors intend to create a systematic review dedicated to the investigation of sources, form of presentation and moments of providing feedback for novice programmers. Another work that is also in progress is a systematic review to investigate how much the feedback provided to novice programmers is in line with the idea of computational thinking Wing (2006).

References

Ahmed, U.Z., Kumar, P., Karkare, A., Kar, P., Gulwani, S. (2018). Compilation error repair: for the student programs, from the student programs. In: *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, pp. 78–87.

Ahmed, U.Z., Sindhgatta, R., Srivastava, N., Karkare, A. (2019). Targeted example generation for compilation errors. In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 327–338. IEEE.

Ahmed, U.Z., Srivastava, N., Sindhgatta, R., Karkare, A. (2020). Characterizing the pedagogical benefits of adaptive feedback for compilation errors by novice programmers. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, pp. 139–150.

Al-Imamy, S., Alizadeh, J., Nour, M.A. (2006). On the development of a programming teaching tool: The effect of teaching by templates on the learning process. *Journal of Information Technology Education: Research*, 5(1), 271–283.

Alwabel, A. (2021). CoEdit: A novel error correction mechanism in compilers using spelling correction algorithms. *Journal of King Saud University-Computer and Information Sciences*.

Barnes, T., Stamper, J. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. In: *International Conference on Intelligent Tutoring Systems*, pp. 373–382. Springer.

Becker, B.A., Goslin, K., Glanville, G. (2018). The effects of enhanced compiler error messages on a syntax error debugging test. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 640–645.

Belcadhi, L.C. (2016). Personalized feedback for self assessment in lifelong learning environments based on semantic web. *Computers in Human Behavior*, 55, 562–570.

Benotti, L., Aloi, F., Bulgarelli, F., Gomez, M.J. (2018). The effect of a Web-based coding tool with automatic feedback on students' performance and perceptions. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 2–7.

Bhatia, S., Kohli, P., Singh, R. (2018). Neuro-symbolic program corrector for introductory programming assignments. In: *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pp. 60–70. IEEE.

Boud, D., Falchikov, N. (2007). *Rethinking Assessment in Higher Education: Learning for the Longer term*. Rout-ledge.

Brusilovsky, P., Weber, G. (1996). Collaborative example selection in an intelligent example-based programming environment.

Butler, D.L., Winne, P.H. (1995). Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research*, 65(3), 245–281.

Carless, D., Salter, D., Yang, M., Lam, J. (2011). Developing sustainable feedback practices. *Studies in Higher Education*, 36(4), 395–407.

Caspersen, M.E., Bennedsen, J. (2007). Instructional design of a programming course: a learning theoretic approach. In: *Proceedings of the Third International Workshop on Computing Education Research*, pp. 111–122.

Cavalcanti, A.P., Diego, A., Mello, R.F., Mangaroska, K., Nascimento, A., Freitas, F., Gašević, D. (2020). How good is my feedback? a content analysis of written feedback. In: *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pp. 428–437.

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215.

Chow, S., Yacef, K., Koprinska, I., Curran, J. (2017). Automated data-driven hints for computer programming students. In: *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pp. 5–10.

Combéfis, S., Schils, A. (2016). Automatic programming error class identification with code plagiarism-based clustering. In: *Proceedings of the 2nd International Code Hunt Workshop on Educational Software Engineering*, pp. 1–6.

Corbett, A.T., Anderson, J.R. (2001). Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 245–252.

Day, M., Penumala, M.R., Gonzalez-Sanchez, J. (2019). Annete: An intelligent tutoring companion embedded into the eclipse IDE. In: *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 71–80. IEEE.

Denny, P., Prather, J., Becker, B.A., Mooney, C., Homer, J., Albrecht, Z.C., Powell, G.B. (2021). On designing programming error messages for novices: Readability and its constituent factors. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–15.

Edmison, B., Edwards, S.H. (2020). Turn up the heat!: Using heat maps to visualize suspicious code to help students successfully complete programming problems faster. In: *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 34–44. IEEE.

Edwards, S.H., Murali, K.P. (2017). CodeWorkout: short programming exercises with built-in data collection. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 188–193.

Endres, M., Sakkas, G., Cosman, B., Jhala, R., Weimer, W. (2019). InFix: automatically repairing novice program inputs. In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 399–410. IEEE.

Fabic, G.V.F., Mitrovic, A., Neshatian, K. (2018). Adaptive problem selection in a mobile Python tutor. In: *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pp. 269–274.

Feldman, M.Q., Wang, Y., Byrd, W.E., Guimbretière, F., Andersen, E. (2019). Towards answering "Am I on the right track?" automatically using program synthesis. In: *Proceedings of the 2019 ACM SIGPLAN Symposium on SPLASH-E*, pp. 13–24.

Friedman, H.S. (2015). *Encyclopedia of mental health*.

Gao, J., Pang, B., Lumetta, S.S. (2016). Automated feedback framework for introductory programming courses. In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 53–58.

Glassey, R. (2019). Developing feedback analytics: Discovering feedback patterns in an introductory course. In: *Proceedings of the ACM Conference on Global Computing Education*, pp. 37–43.

Gusukuma, L., Bart, A.C., Kafura, D., Ernst, J. (2018). Misconception-driven feedback: Results from an experimental study. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*, pp. 160–168.

Hajja, A., Hunt, A.J., McCauley, R. (2019). PolyPy: A Web-Platform for Generating Quasi-Random Python Code and Gaining Insights on Student Learning. In: *2019 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8. IEEE.

Haldeman, G., Tjang, A., Babeş-Vroman, M., Bartos, S., Shah, J., Yucht, D., Nguyen, T.D. (2018). Providing meaningful feedback for autograding of programming assignments. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 278–283.

Hameer, A., Pientka, B. (2019). Teaching the art of functional programming using automated grading (experience report). *Proceedings of the ACM on Programming Languages*, 3(ICFP), 1–15.

Hattie, J., Timperley, H. (2007). The power of feedback. *Review of Educational Research*, 77(1), 81–112.

Holmes, L.E., Smith, L.J. (2003). Student evaluations of faculty grading methods. *Journal of Education for Business*, 78(6), 318–323.

Höppner, F. (2019). Measuring instruction comprehension by mining memory traces for early formative feedback in Java courses. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 105–111.

Indriasari, T.D., Luxton-Reilly, A., Denny, P. (2021). Investigating Accuracy and Perceived Value of Feedback in Peer Code Review Using Gamification. In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pp. 199–205.

Jemmali, C., Kleinman, E., Bunian, S., Almeda, M.V., Rowe, E., Seif El-Nasr, M. (2020). MAADS: Mixed-Methods Approach for the Analysis of Debugging Sequences of Beginner Programmers. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pp. 86–92.

Jessop, T., El Hakim, Y., Gibbs, G. (2014). The whole is greater than the sum of its parts: a large-scale study of students' learning in response to different programme assessment patterns. *Assessment & Evaluation in Higher Education*, 39(1), 73–88.

Jiang, L., Rewcastle, R., Denny, P., Tempero, E. (2020). CompareCFG: Providing Visual Feedback on Code Quality Using Control Flow Graphs. In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 493–499.

Kadekar, H.B., Sohoni, S., Craig, S.D. (2018). Effects of error messages on students' ability to understand and fix programming errors. In: *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8. IEEE.

Keuning, H., Jeuring, J., 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*, pp. 41–46.

Keuning, H., Jeuring, J., Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)*, 19(1), 1–43.

Kitchenham, B., Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Knight, P., Yorke, M. (2003). Assessment, learning and employability.

Koulouri, T., Lauria, S., Macredie, R.D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE)*, 14(4), 1–28.

Kumar, A.N. (2006). Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors. *Technology, Instruction, Cognition and Learning (TICL) Journal*, 4(1), 65–107.

Kunkle, W.M., Allen, R.B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1), 1–26.

Kyrilov, A., Noelle, D.C. (2016). Do students need detailed feedback on programming exercises and can automated assessment systems provide it? *Journal of Computing Sciences in Colleges*, 31(4), 115–121.

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14–18.

Langer, P. (2011). The use of feedback in education: a complex instructional strategy. *Psychological Reports*, 109(3), 775–784.

Latih, R., Bakar, M.A., Jailani, N., Ali, N.M., Salleh, S.M., Zin, A.M. (2017). PC 2 to support instant feedback and good programming practice. In: *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 1–5. IEEE.

Liu, X., Wang, S., Wang, P., Wu, D. (2019). Automatic grading of programming assignments: an approach based on formal semantics. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 126–137. IEEE.

Lobb, R., Harlow, J. (2016). Coderunner: A tool for assessing computer programming skills. *ACM Inroads*, 7(1), 47–51.

Loksa, D., Ko, A.J., Jernigan, W., Oleson, A., Mendez, C.J., Burnett, M.M. (2016). Programming, problem solving, and self-awareness: Effects of explicit guidance. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 1449–1461.

Luxton-Reilly, A., Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., 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*, pp. 55–106.

Marwan, S., Jay Williams, J., Price, T. (2019). An evaluation of the impact of automated programming hints on performance and learning. In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*, pp. 61–70.

Marwan, S., Gao, G., Fisk, S., Price, T.W., Barnes, T. (2020). Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In: *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pp. 194–203.

Mathew, R., Malik, S.I., Tawafak, R.M. (2019). Teaching Problem Solving Skills using an Educational Game in a Computer Programming Course. *Informatics in Education*, 18(2), 359–373.

Medeiros, R.P., Ramalho, G.L., Falcão, T.P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2), 77–90.

Mendoza, B., Reyes-Alamo, J., Wu, H., Carranza, A., Zavala, L. (2016). iPractice: a self-assessment tool for students learning computer programming in an urban campus. *Journal of Computing Sciences in Colleges*, 31(3), 93–100.

Mory, E.H. (2004). Feedback research revisited.

Mutch, A. (2003). Exploring the practice of feedback to students. *Active Learning in Higher Education*, 4(1), 24–38.

Nguyen, A., Piech, C., Huang, J., Guibas, L. (2014). Codewebs: scalable homework search for massive open online programming courses. In: *Proceedings of the 23rd International Conference on World Wide web*, pp. 491–502.

Nicol, D.J., Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2), 199–218.
Orrell, J. (2006). Feedback on learning achievement: rhetoric and reality. *Teaching in Higher Education*, 11(4), 441–456.
Ott, C., Robins, A., Shephard, K. (2016). Translating principles of effective feedback for students into the CS1 context. *ACM Transactions on Computing Education (TOCE)*, 16(1), 1–27.
Parikh, A., McReelis, K., Hodges, B. (2001). Student feedback in problem based learning: a survey of 103 final year students across five Ontario medical schools. *Medical Education*, 35(7), 632–636.
Perera, P., Tennakoon, G., Ahangama, S., Panditharathna, R., Chathuranga, B. (2021). A Systematic Review of Introductory Programming Languages for Novice Learners. *IEEE Access*.
Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. *ACM SIGCSE Bulletin*, 35(2), 78–82.
Poulos, A., Mahony, M.J. (2008). Effectiveness of feedback: The students' perspective. *Assessment & Evaluation in Higher Education*, 33(2), 143–154.
Qian, Y., Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1–24.
Renzella, J., Cain, A. (2020). Enriching programming student feedback with audio comments. In: *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 173–183. IEEE.
Rust, C., Price, M., O'DONOVAN, B. (2003). Improving students' learning by developing their understanding of assessment criteria and processes. *Assessment & Evaluation in Higher Education*, 28(2), 147–164.
Sadler, D.R. (1989). Formative assessment and the design of instructional systems. *Instructional Science*, 18(2), 119–144.
Schunk, D.H., Zimmerman, B.J. (1998). *Self-regulated Learning: From Teaching to Self-reflective Practice*. Guilford Press, New York.
Sheard, J., Simon, S., Hamilton, M., Lönnberg, J. (2009). Analysis of research into the teaching and learning of programming. In: *Proceedings of the fifth International Workshop on Computing Education Research Workshop*, pp. 93–104.
Shute, V.J. (2008). Focus on formative feedback. *Review of educational research*, 78(1), 153–189.
Smith, R., Tang, T., Warren, J., Rixner, S. (2017). An automated system for interactively learning software testing. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 98–103.
Smith, R., Tang, T., Warren, J., Rixner, S. (2019). Auto-generating visual exercises for learning program semantics. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 360–366.
Spohrer, J.C., Soloway, E. (1989). Simulating student programmers. In: *IJCAI* (Vol. 89), pp. 543–549. Citeseer.
Staubitz, T., Klement, H., Teusner, R., Renz, J., Meinel, C. (2016). CodeOcean-A versatile platform for practical programming excercises in online environments. In: *2016 IEEE Global Engineering Education Conference (EDUCON)*, pp. 314–323. IEEE.
Stephens-Martinez, K., Fox, A. (2018). Giving hints is complicated: understanding the challenges of an automated hint system based on frequent wrong answers. In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 45–50.
Treviño, Y.M., Cavazos, M.R.L. (2018). Effects of immediate feedback using ICT in a CS1 course that implements Mastery Learning. In: *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5. IEEE.
UNESCO (2017). Education for sustainable development goals: Learning objectives.
Veerasamy, A.K., D'Souza, D., Lindén, R., Laakso, M.-J. (2019). Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses. *Journal of Computer Assisted Learning*, 35(2), 246–255.
Wang, K., Singh, R., Su, Z. (2018). Search, align, and repair: data-driven feedback generation for introductory programming exercises. In: *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 481–495.
Wang, W., Rao, Y., Zhi, R., Marwan, S., Gao, G., Price, T.W. (2020). Step tutor: Supporting students through step-by-step example-based feedback. In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 391–397.
Weaver, M.R. (2006). Do students value feedback? Student perceptions of tutors' written responses. *Assessment & Evaluation in Higher Education*, 31(3), 379–394.
Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

Yan, Y.-X., Wu, J.-P., Nguyen, B.-A., Chen, H.-M. (2020). The impact of iterative assessment system on programming learning behavior. In: *Proceedings of the 2020 9th International Conference on Educational and Information Technology*, pp. 89–94.

Zimmerman, B.J. (2013). From cognitive modeling to self-regulation: A social cognitive career path. *Educational Psychologist*, 48(3), 135–147.

H.J. Barbosa Rocha is a teacher of Informatics at Federal Institute of Alagoas, in a rural school in the northeast of Brazil. She received a B.S. degree in Information System from the Faculty of Alagoas (FAL), Brazil, in 2009, an M.Sc. degree Knowledge Computational Modeling in from the Federal University of Alagoas, Brazil, in 2012. Currently, she is a PhD student at Federal University of Pernambuco, Brazil, working on adaptive and personalized feedback for the programming domain. Her research interests include Intelligent Tutoring Systems, Student Modeling and Adaptive Feedback, as well as Knowledge Representation and Reasoning, Personalized Recommender Systems.

P. Cabral de Azevedo Restelli Tedesco is an Associate Professor in Computer Science at the Centro de Informática of the Federal University of Pernambuco – UFPE. She is has got her undergraduate degree from UFPE in 1994, ther MSc from the same university in 1997 and her PhD from the Computer Based Learning Unit-University of Leeds, where she has worked under the supervision of Prof. John Self. She has experience in the Computer Science Field, with a focus on Artificial Intelligence applied to Education, having published more than a 100 papers and supervised many graduate students in the field.

E. de Barrros Costa is Professor at Federal University of Alagoas, Brazil. He received a B.S. degree in Computer Science from Federal University of Paraiba (UFPB), Brazil in 1988, an M.Sc. and a Ph.D degree in Electrical Engineering from UFPB in 1989 and 1997. His research interests include Intelligent Tutoring Systems, knowledge representation and reasoning, personalized recommendation systems, machine learning and multiagent systems.