A Systematic Literature Review of Methodology of Learning Evaluation Based on Item Response Theory in the Context of Programming Teaching

Jucelio S. Santos*, Wilkerson L. Andrade*, João Brunet*, Monilly Ramos Araujo Melo*
*Federal University of Campina Grande (UFCG), Campina Grande, Brazil
jucelio@copin.ufcg.edu.br, {wilkerson, joao.arthur}@computacao.ufcg.edu.br, monillyramos@gmail.com

***Abstract*—This Research Full Paper presents a Systematic Literature Review (SLR) that investigates state-of-the-art methods, processes, approaches, and instruments based on Item Response Theory (IRT) in the programming teaching context. Various studies support professors and students to improve the evaluation process in teaching programming. Among the different techniques and theories associated with these studies, the IRT gained prominence because of its effectiveness. Due to the lack of an overview in the area, we present a SLR with the objective of identity studies that use IRT as a methodology of evaluation of learning in the context of Programming Teaching; studies which present real-time feedback; and experimental studies that have been carried out to validate them. To achieve these goals, we planned an SLR following the guidelines proposed by Kitchenham (2004). Our main findings are a) There is a limited number of studies that explore IRT in evaluation methodologies in the teaching programming context. Among these studies, the main focus was the development on of instruments to measure programming skills; b) Most instruments are multiple-choice, adopt the 3PL model, and do not show evidence of real-time feedback except SIETTE and CodeWorkout and are also the only instruments that evaluate the coding capacity of individuals; c) The studies showed scientific evidence on the use of IRT in the evaluation methodology in the programming teaching context. The studies presented experiments that indicate a significant gain in the measurement of programming abilities when compared to the traditional evaluation method.**

I. Introduction

Traditionally, programming teaching is difficult to assimilate, based on theoretical lessons, presentation of pseudocode examples, and later problem-solving. The focus on a series of commands and codes of a specific programming language, selecting exercises as an example, usually presenting them in a non-practical context, which makes it difficult to create mental images and models that help in the construction of a language [1]. The assessment of student learning is another issue that is part of the scope of the traditional teaching process. A test diagnoses the mental state of each student in the subject. The fact that a student achieves average performance in a programming course does not indicate that the student has learned half of the content, possibly learned one topic very well, and almost nothing in other related topics [2].

In the academy, there are ways to overcome these limitations. Various methods, processes, approaches, and tools have been developed to support teachers and students to improve the assessment process in programming teaching. Among the different techniques and theories associated with these studies, Item Response Theory (IRT) has gained projection for its effectiveness.

IRT is a statistical theory used by psychometrics and the educational area for the construction, evaluation, and validation of instruments. The mathematical models involved depending on the adopted logistic model and the dimension of the instrument. Although other areas have analyzed the evaluative potential of IRT, there is still very little understanding about the impacts and contributions that IRT offers considering methods, processes, approaches, and instruments in the context of programming teaching.

In this context, it is necessary to investigate the best evaluation practices that use IRT associated with programming teaching. The objective of this paper is to identify studies that use IRT as a methodology for the assessment of learning in the context of programming teaching, as well as to list studies that present real-time evaluative feedback and which experimental studies were performed to validate them. To achieve these goals, we planned and conducted a Systematic Literature Review (SLR) following the guidelines proposed by Kitchenham et al. [3].

We organized this paper as follows. In Section II, we briefly discuss related work. In Section III, we present the methodology applied to conduct in this paper. In section IV, we offer the results and their discussions in section V. Finally; in Section VI, we present the conclusions and suggestions for future work.

II. Related Work

To the best of our knowledge, there is no SLR in the methodology for learning assessment that adopts IRT in the context of programming teaching. Existing studies address the challenges of introductory programming teaching/assessment by adopting alternative methods, skills and formative feedback [4] [5] [6]. These studies are important because they provide an overview of essentials in introductory programming that teach and indicate the skills that should be promoted and evaluated.

The SLR presented by Medeiros et al. [7] characterized the challenges of introductory programming teaching. It highlighted the fundamental skills for a novice student to learn to program, as well as the difficulties they faced in this process and outlined the challenges teachers encounter in

teaching introduction to programming. The main teaching challenge pointed out by the authors concerns the lack of appropriate methods and tools for personalized teaching. They cited problem-solving as a learning challenge, followed by motivation and engagement, and difficulty learning the syntax of programming languages. We present the methodology applied to conduct in this paper.

Teaching beginners programming is a difficult task because of the complex nature of the subject, as negative stereotypes are associated with programming. The introductory programming courses often do not encourage student understanding. The systematic review presented by Major et al. [8] has investigated the effectiveness of using robots as tools to aid the programming teaching process, even though robots can be a powerful and useful tool when used in an introductory programming course. Still, the potential remains to investigate further methods for their implementation.

Formative feedback, designed to help students improve their work, is an essential factor in learning. Many tools offer programming exercises that provide automated feedback on student solutions. Another SLR [6] provides an overview of what types/techniques of feedback are used by these tools, most of which identify errors to correct the problem. Also, teachers cannot quickly adapt machines to their own needs. However, the diversity of feedback types has increased in recent decades, and new techniques are being applied to generate feedback that is increasingly useful to the student.

In this study, we presented SLR to identify studies that explore IRT as a methodology of evaluation of learning in the context of Programming Teaching. Studies that present real-time feedback and experimental studies that have been carried out to validate them.

III. Research Methodology

We performed an SLR to obtain knowledge about the use of IRT in the context of programming teaching, following the guidelines proposed by Kitchenham [3]. Through an SLR, we can identify, evaluate, and interpret all the researches that are relevant to a particular research question or thematic area or an example of interest repetitively or impartially.

SLR differs from a review of the literature by the definition of the evaluation protocol. The data obtained from SLR are prerequisites for a quantitative meta-analysis. First, we defined the review protocol that allowed us to identify the objectives, the research questions, the scope, the strategy search, and the search query selection studies, as presented below.

A. Objectives and scope

Although the initial research interest is related to introductory programming discipline, we noted that with this scope limitation, many relevant works could not be found. Thus, we have broadened the scope of this SLR and defined the following objectives:

- **Objective 1:** identify methods, processes, approaches, and instruments that use IRT as a methodology of evaluation of learning in the context of Programming Teaching;
- **Objective 2:** identify studies which present real-time feedback;
- **Objective 3:** identify experimental studies that have been carried out to validate them.

Table I
Terms, keywords and synonyms used to create the search query

Term	Keywords	Synonyms
A	Item Response Theory	"Latent Trait Theory", "Strong True Score Theory", "Modern Mental Test Theory"
B	Introduction to Programming	"Programming Course", "Programming Language", "Programming Learning", "Learning Programming", "Programming Teaching", "Teaching Programming"

B. Research questions

To meet the proposed objectives, we divided the research question (RQ) of this SLR as follows:

- **Primary question (PQ):** what methods, processes, approaches, and instruments use IRT to measure programming skills?
- **Secondary question (SQ1):** do the studies provide real-time feedback?
- **Secondary question (SQ2):** how do researchers validate the studies?

C. Search strategy

Our search strategy consisted of an online search in the three main digital libraries with high relevance for Software Engineering, namely: IEEE Xplore [9], ACM Digital Library [10], and Science Direct [11].

In online digital libraries, search keywords are essential for the quality and coverage of results, so they must be defined carefully. The query must be searched using a necessary sequence, consisting of two terms, one standard term and the other a combination of synonyms for the identified keywords. We present the keywords and synonyms for each term in Table I. The first term (A) refers to IRT, and the second term (B) refers to Introduction to Programming and its synonyms.

It is essential to mention that for each digital library, we perform full-text search and metadata (title, keyword, and abstract) of the publications. To evaluate the search query, we performed a pilot search on the IEEE digital library. This evaluation consisted of verifying that the main related works were returned; we presented the search sequence for each digital library in Table II.

D. Study selection strategy

We explicitly set the inclusion and exclusion criteria in reviewing the protocol for this SLR. The primary studies included should fall into one of three inclusion criteria:

- **Inclusion criteria (IC1):** studies that define a method, process, approach or instrument that use IRT to evaluate/measure programming skills;

Table II
SEARCH QUERY IN DIGITAL LIBRARIES

Digital Library	Search Query
IEEE	((“Item Response Theory” OR “Latent Trait Theory” OR “Strong True Score Theory” OR “Modern Mental Test Theory”) AND (“Introduction to Programming” OR “Programming Course” OR “Programming Language” OR “Programming Learning” OR “Learning Programming” OR “Programming Teaching” OR “Teaching Programming”))
ACM	content.ftsec:(“Item Response Theory” OR “Latent Trait Theory” OR “Strong True Score Theory” OR “Modern Mental Test Theory”) AND (“Introduction to Programming” OR “Programming Course” OR “Programming Language” OR “Programming Learning” OR “Learning Programming” OR “Programming Teaching” OR “Teaching Programming”)
Science Direct	((“Item Response Theory” OR “Latent Trait Theory” OR “Strong True Score Theory” OR “Modern Mental Test Theory”) AND (“Introduction to Programming” OR “Programming Course” OR “Programming Language” OR “Programming Learning” OR “Learning Programming” OR “Programming Teaching” OR “Teaching Programming”))

- **Inclusion criteria (IC2):** studies which present real-time feedback;
- **Inclusion criteria (IC3):** experimental studies that have been carried out to validate them.

We considered some exclusion criteria in the SLR, namely:

- **Exclusion criteria (EC1):** studies is not written english;
- **Exclusion criteria (EC2):** studies is a short paper or SLR;
- **Exclusion criteria (EC3):** incomplete studies, unavailable and/or duplicate studies (papers with the same or updated version, keeping only the most recent);
- **Exclusion criteria (EC4):** studies that do not define a method, process, approach or instrument that use IRT to evaluate/measure programming skills;

We organized our study selection process in four distinct phases, described below:

- **Phase 1:** as a preliminary selection, we performed the queries, applied EC4 and defined the study group that served for the second phase;
- **Phase 2:** based on the titles, abstracts, and words of the preliminary selection studies, we determined and kept which were studies relevant;
- **Phase 3:** based on the inclusion and exclusion criteria and full reading, we reviewed relevant studies from the previous phase;
- **Phase 4:** one specialist evaluated and validated the selected studies, with the possibility of inclusion or exclusion of studies.

E. Data extraction

The data extraction is intended to summarize data from selected primary studies. We have prepared a form to extract and synthesize relevant study data to answer the research questions defined in the SLR protocol. We set the items and their descriptions, and we presented in Table III.

Table III
DATA EXTRACTION FORM POINTS AND DESCRIPTIONS

Common items	Descriptions
Title	The title of the primary study
Year	The year the primary study was applied
Source	The conference, journal, or book in which the primary study was published
Type	Is it a method, process, approach or instrument?
Used models	Which logistics model does the proposed solution use?
Tools	What tools were used in the study?
Benefits	Benefits of using the proposed solution
Limitations	Limitations of the proposed solution
Feedback	Does the method, process, approach or instrument present immediate feedback?
Validation	How was the proposed solution validated?

The SLR process defined in the previous section resulted in 126 articles found in the three databases. After Phase 1, we selected 36 studies for Phase 2, in which we identified 32 studies [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] as relevant after a careful reading of the article. In the works in which the principal researcher had difficulties in accepting or not the article in the inclusion criteria, we consulted university professors specialized in teaching programming to solve these doubts. We show the details of the study in Table IV by the library and Table V by phase.

Table IV
DETAILS OF STUDY SEARCH AND SELECTION BY DATABASE

Digital Library	Search result
IEEE Xplore	42
ACM Digital Library	12
Science Direct	72
Total studies	**126**
Excluded	**90**
Studies for Phase 2	**36**

Table V
SEARCH PHASE STUDY DETAILS

Phase	Descriptions	Included	Excluded
Phase 1	Search results	36	90
Phase 2	Title and abstract selection	36	0
Phase 3	Full reading selection	32	4
Phase 4	Selection validated by a specialist	32	0

Among these studies, 32 falls into IC1 [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41]

[42] [43], 6 belongs to IC2 [12] [13] [14] [15] [16] [39], and 21 belongs to IC3 [12] [13] [14] [18] [19] [20] [21] [22] [23] [24] [25] [26] [29] [32] [36] [37] [38] [40] [41] [42] [43].

IV. Results

This section details the SLR result. We surveyed from April to July 2019. First, we presented overall results extracted directly from the paper headings, such as countries where the surveyed, publication vehicle (magazine or conference), and distribution studies by year of publication. In the remaining sections, we answer each research question defined in Section III.

A. General results

1) *Countries distribution:* Figure 1 shows the countries distribution where each research has been done. The United States and Spain are the leading countries where methods, processes, approaches, and instruments have been investigated (ten studies). After that, Germany and Slovakia have three studies, Serbia, Marocco, Japan, and Canada have two studies, and other countries like UK, South Korea, Romania, Norway, Malaysia, Colombia, China, and Brazil have only one study. Spain is one of the leading countries in some publications. A possible explanation is given by the large volume of scientific study publications [12] [13] [14] [15] [16] on the SITETTE instrument.
2) *Publication vehicle:* Figure 2 presents the publication vehicle adopted by the selected studies. Conference papers are the most frequent type, with 59.37% (19 studies out of 32). Journals appear with 34.37% (11 studies out of 32), and Magazine article (2 studies out of 32) with 6.25%.
3) *Publication years:* In total, we had studies that were published from 2005 to 2019. Figure 3 shows the distribution of these studies per year. Between the years 2005 and 2009, the studies focused on the construction of multiple-choice instruments to facilitate the process of measuring students' abilities in TRI-based programming. Only in the last ten years has there been a variety of studies that define not only instruments but also methods, processes, and approaches.
4) *Classification of research approaches:* The studies were also classified based on the research approaches. The research approaches chosen by Petersen et al. [44] were as follows: validation research, evaluation research, solution proposal, experiment articles, philosophical articles, and opinion articles. Figure 4 shows the result of distribution studies by research approaches.

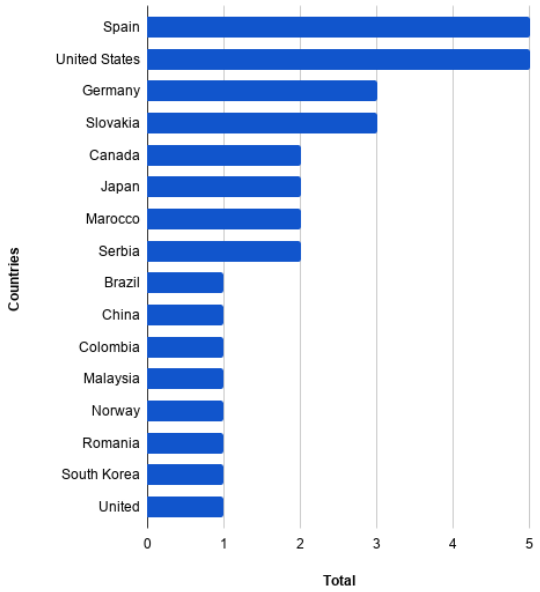

Figure 1. Distribution of studies by countries

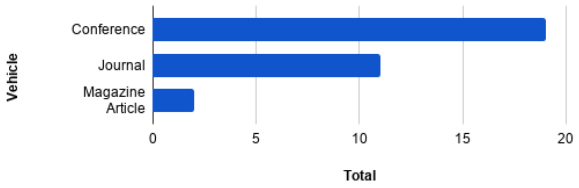

Figure 2. Distribution of studies by publication vehicle

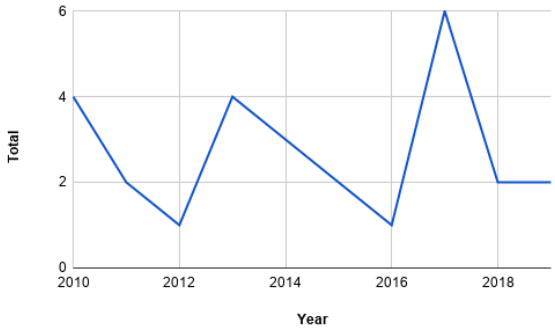

Figure 3. Distribution of studies by year

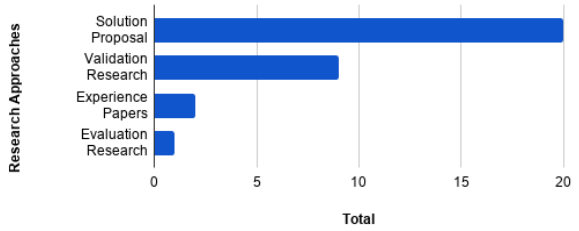

Figure 4. Distribution of studies by research approaches

B. PQ: What methods, processes, approaches, and instruments use IRT to measure programming skills?

Among these studies, we classified three as methods and process, seven as approaches, and 22 define instruments, and some of these studies report the same instrument. In Figure 5, we present the distribution of these studies by category.

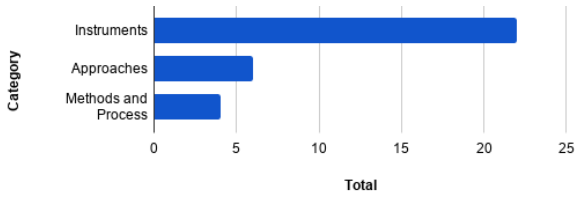

Figure 5. Distribution of studies by category

1) *Methods and process:* Among the primary studies, there have been few 3 papers [17] [18] [19] that have reported on methods and processes for measuring programming skills through IRT.
2) *Approaches:* There have been 7 papers [20] [21] [22] [23] [24] [25] [26] who reported approaching IRT to measure programming skills.
3) *Instruments:* Among the studies, 22 papers [27] [28] [29] [30] [31] [32] [18] [33] [15] [16] [14] [12] [13] [34] [35] [36] [37] [38] [39] [40] [41] [42] define an instrument to measure the ability of programming through IRT.

C. SQ1: Do the studies provide real-time feedback?

In Figure 6, we present the distribution of these studies that provide real-time feedback.

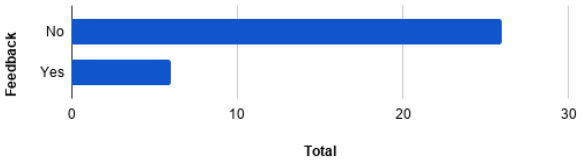

Figure 6. Distribution of studies that provide real-time feedback

1) *Methods and process:* All the studies that have reported a method or a process [17] [43] [19] of the IRT to measure programming skills did not present characteristics related to real-time feedback.
2) *Approaches:* Even an approach [20] [21] [22] [24] [25] [26] of the IRT to measure programming skills did not present characteristics related to real-time feedback. In general, these studies have explored the IRT to calibrate items, estimate the ability of a group of individuals, identify a set of competencies, and other characteristics that are useful for obtaining information for both the teacher and the student, the app.
3) *Instruments:* The studies [18] [33] [30] [40] [38] [37] [36] [42] [41] did not show evidence about the use of feedback in their instruments.

D. SQ2: How do researchers validate the studies?

In Figure 7, we present the distribution of these validated studies.

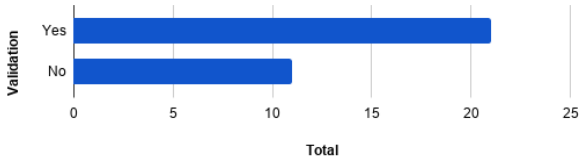

Figure 7. Distribution of validated studies

1) *Methods and process:* Among the methods and processes, the studies [43] [19] present validation in their proposals.
2) *Approaches:* All studies [26] [20] [24] [22] [25] [23] [21] that addressed IRT to measure programming skills presented experiments as a way to validate them.
3) *Instruments:* Among the instruments, the following studies [14] [12] [13] [18] [38] [37] [32] [41] [29] [42] [36] [40] presented scientific evidence of its efficiency through empirical studies.

Other studies only report a method/process [17] or instrument [30] [39] [28] [15] [16] [27] [34] [35] [31] [33] present no scientific evidence of its efficiency, some of which are proven in future studies.

V. Discussions

This section discusses the data presented in Section IV on the methods, processes, approaches, and instruments that use IRT to measure programming skills, evidencing studies that approach immediate feedback and the existing empirical studies that demonstrate its validity.

A. Methods and processes

Acquiring programming skills is a challenging task. Suppose a group of students begins to learn to program without computer skills. They need to learn many different new terminologies and various types of knowledge to run even an elementary program. Thus, it is necessary to offer a promising educational method that encourages students to program, as well as refine it all the time, thus requiring a high cost.

Shimakawa et al. [17] proposed a new method for refining the programming training course. This method uses measures to evaluate the content of teaching in the class. The method reduces the cost of refinement due to its collective intelligence approach.

Another essential factor that we must observe in the teaching of programming is the analysis of the source code. This analysis provides exciting information about learning progress, especially for beginning programmers or introductory programming courses. Analyzing the ability to code is a very complex task and difficult to observe directly in its entirety. For this purpose, it is possible to use IRT [43] to assess this ability.

Thus, Berges and Hubwieser [43] proposed a method for assessing coding skills and demonstrated it during programming projects, showing results that strengthen new information about the difficulty of applying concepts and the distribution of coding skills from the students.

In parallel, Oeda and Kosaku [19] proposed a new code review method using check sheets to evaluate the skills of novice programmers better. This check sheet is designed by choosing from several items to observe programming lessons. The method uses the IRT to estimate the level of difficulty and discrimination of the items present in the exam. From the experimental results, it was possible to visualize the parameters and the information function of the items, as well as the estimation of the student's skill level.

B. Approaches

Since the early Programmer for International Student Assessment (PISA) research, the intentions of education are increasingly expressed in terms of skills, rather than learning objectives. As a consequence, learning outcomes should also be measured in terms of the intended competencies. This applies to large-scale investigations such as PISA, as well as to the very small scale represented by the examinations that a single teacher performs with his students.

However, to ensure validity, competency measurement requires adequate models for structure and competence levels, based on empirical research.

Among the studies of this SLR, some approaches were conducted and expressed in terms of skills. These studies used IRT to measure these programming skills: Computational Thinking [26] [25]; Object-oriented programming [20] and Python [21].

Some studies took advantage of data provided by online programming judge systems and estimated instrument item parameters and individuals' abilities through IRT [22] [23] [24].

C. Instruments

Among the studies, most of them aim to define an instrument to measure programming capacity through IRT. According to Table VI, among the instruments, measures basic programming skills (28.6%), multilingual (21.4%) Java (21.4%), C ++ (7.1%), object-orientation (7.1)%)), Lisp (7.1%) and Computational Thinking (7.1%).

Besides that, 64.3% of them adopted 3 Parameter Logistic Model (3PL), 7.1% adopted the 2 Parameter Logistic Model (2PL), and 28.6% considered 1 Parameter Logistic Model (1PL). IRT presents three logistical models. The 3PL considers the parameters of difficulty, discrimination, and the probability of hitting the item by chance. The 2 PL considers the item's discrimination and difficulty parameters. Moreover, the 1PL evaluates only the difficulty of the item and is also known as the Rasch model. The choice for one of these models depends on the fit of the data collected from the real world to the model [**?**].

Table VI
COMPARISON BETWEEN INSTRUMENTS

Name	Model	Language/Paradigm	Study
The University of Hertfordshire	3PL	Programming	[27]
Flip	3PL	Lisp	[28] [29]
The Universidad Distrital Francisco José de Caldas	3PL	Object orientation	[30]
GRE (Graduate Record Exam)	3PL	C++	[31] [32]
The University of Oslo	1PL	Java	[18]
The University Tun Hussein Onn Malaysia	1PL	Programming	[33]
SIETTE	3PL	Multilanguage	[15] [16] [14] [12] [13]
MSA-IDUII	2PL	Java	[34] [35]
The Cheju National University	3PL	Programming	[36]
FCS1	3PL	Multilanguage	[37] [38]
Code Workout	3PL	Multilanguage	[39]
The mathematics and informatics department	3PL	Java	[40]
The University of Miami	1PL	Computational Thinking	[41]
The TUM School of Education Technische Universitat Munchen	1PL	Programming	[42]

The ideal learning process is one where students can receive classes, resolve exercises, and obtain immediate feedback from the teacher [14]. However, providing real-time feedback from the teacher is an impossible task in the face of natural conditions.

For example, in a classroom of 30 students, it is impracticable for a teacher to be able to get the students' doubts about the problem-solving process (coding). In this way, some digital instruments contribute to this process, making it automatic.

Among the instruments available, System of Intelligent Evaluation Using Tests for Tele Education (SIETTE) [15] [16] [14] [12] [13] and CodeWorkout [39] provide real-time feedback.

SIETTE is a web-based assessment environment that arises from an attempt to merge CATs (Computerized Adaptive Tests) and web technology. The instrument measures various skills in programming the IRT medium. The environment also acts as a compiler, letting users know if their solution is syntactically correct or not. The system shows a list of all violated restrictions and detailed feedback, which will be very specific or more general, depending on the student's estimated level.

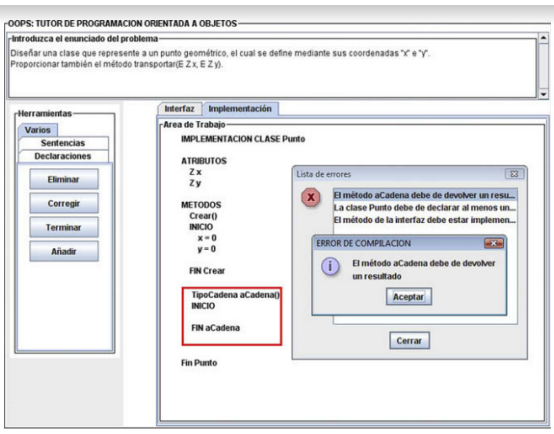

Figure 8. Feedback from an SIETTE instrument item

Several SIETTE empirical studies report improvements in the overall reliability of the validity and quality of each item in the instrument, as well as estimates of students' level of knowledge, instantiating interventional methods that improve students' performance on final exams [14] [12] [13].

CodeWorkout [39] is a set of code practices and training for short programming exercises and questions of choice. CodeWorkout combines an open and gradual knowledge model that allows any individual to practice exercises. Figure 9 shows the view students use to enter answers for and receive feedback on exercises. CodeWorkout uses an "exercising" metaphor to reinforce the idea that users are building their strengths (at programming).

Figure 9. Feedback from an Code-Workout instrument item

As a result, it uses terminology based on this metaphor, calling individual problem exercises, while collections of problems are called workouts, and the public practice area is known as the academy. Students who practice on their earn experience points instead of a grade and practice counts towards the system's notion of topic mastery, whether conducted on one's own or as part of a class assignment. However, there are no studies that scientifically prove its effectiveness in estimating programming skills, reporting only students' perceptions of the instrument and its benefits.

It should be noted that some of the instruments present feedback at the end of the resolution of the items: i) The Graduate Record Exam (GRE) [32] [31] is an instrument developed to measure C ++ programming skills; And ii) Measuring Skills Approach using Increment and Decrement Units of Item's Index (MSA-IDUII) to measure Java programming skills; [34] [35]. In both instruments, a dashboard is provided with the individual's ultimate performance (see Fig. 10 and Fig. 11).

Figure 10. GRE Test results view

In their study, Cisar et al. [32] analyzed the data generated from the application of instrument GRE. The results of the research show that the Students who worked on a computerized adaptive test scored a higher average score than students who did the traditional test.

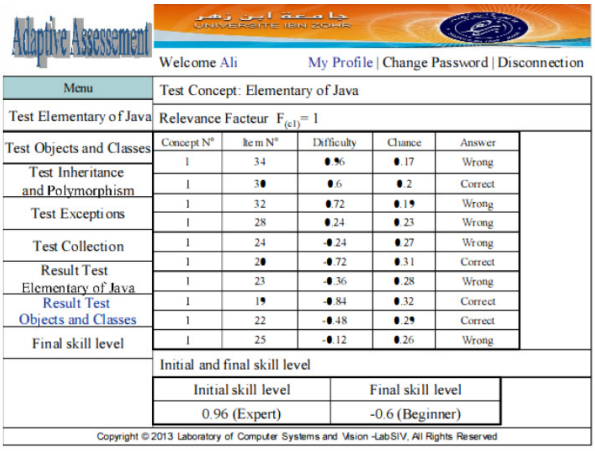

Figure 11. MSA-IDUII Test results view

The University of Hertfordshire Instrument [27] also provides feedback at the end of the resolution of the items. The overall score or general level of proficiency is estimated by the CAT algorithm using the complete set of responses for a given candidate. Figure 12 illustrates how this information is displayed through automated feedback.

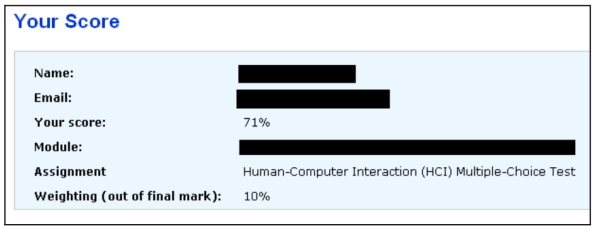

Figure 12. Final Feedback generated after instrument completion

The other instruments explored in this SLR have no real-time feedback, but have provided some evidence on their application, namely: i) the results obtained with the application of the Flip [28] [29] instrument can be used as feedback for recommendations for the next learning topic. ii) The University of Oslo Instrument [18] presents several tests in order to prove its efficiency. iii) The validity of the FCS1 assessment used a multifaceted argument, combining interview data, statistical analysis of assessment results, and CS1 exam scores [37] [38]. iv) The results conducted in the student experiment using The Cheju National University [36] instrument evinced ease in classifying and manipulating learning content as well as categorizing students into various groups, as well as v) The mathematics and informatics department instrument [40]. Finally, vi) The University of Miami instrument [41] and vii) The TUM School of Education Technische Universitat Munchen instrument [42] present good psychometric properties and has the potential to reveal student learning challenges and growth in terms of Computational Thinking and Programming, respectively.

VI. Conclusion and Future Works

This paper aimed to present an SLR to obtain the state of the art of methods, processes, approaches, and instruments based on IRT in the context of teaching programming. We highlight the studies that deal with immediate feedback and verify the existence of empirical studies that demonstrate its validity. For that, we defined the SLR protocol, presented the search, and the results of this review.

As a result, we found that, in recent years, despite the benefits of IRT for the area of psychometric assessment, (PQ) few studies are exploring this theory in Teaching Programming. Among these studies, the main focus was the development of instruments to measure programming skills; (SQ1) most instruments are multiple-choice, adopt the 3PL model and do not show evidence of real-time feedback, except SIETTE and CodeWorkout, and are also the only instruments that assess the coding capacity of individuals; (SQ2) the studies listed in this SLR showed scientific evidence about the use of IRT in the programming learning assessment methodology. The studies presented experiments that indicate a significant gain in the measurement of programming skills when compared to the traditional method of evaluation.

The limitations of this SLR are that, although in the researched libraries, they index the most influential magazines and conferences in this area, we cannot guarantee that the SLR will cover them completely. Therefore, for future work, we intend to extend this SLR to annals of the main conferences in the field of Computing and periodicals.

References

[1] C. E. Rapkiewicz, G. A. M. Falkembach, L. M. J. Seixas, N. S. Santos, S. Rosa, V. V. Cunha, and M Klemann. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE: Revista Novas Tecnologias na Educação*, 2007.

[2] E. P. Pimentel, V. F. de França, R. V. Noronha, and N. Omar. Avaliação contínua da aprendizagem, das competências e habilidades em programção de computadores. In *Proceedings of the Workshop de Informática na Escola (WEI)*. SBC, Porto Alegre, RS, Brazil, 2003.

[3] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33, 2004.

[4] I. Huet, O. R. Pacheco, J. Tavares, and G. Weir. New challenges in teaching introductory programming courses: A case study. In *Proceedings of the IEEE Frontiers in Education Conference (FIE)*. IEEE, 2004.

[5] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang. Ability-training-oriented automated assessment in introductory programming course. *Computers & Education*, 56(1), 2011.

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

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

[8] L. Major, T. Kyriacou, and O. P. Brereton. Systematic literature review: Teaching novices programming using robots. *IET Software*, 6(6), 2012.

[9] Ieee xplore library. https://ieeexplore.ieee.org/. Accessed: 2019-08-16.

[10] Acm digital library. https://dl.acm.org/. Accessed: 2019-08-16.

[11] Sciencedirect library. https://www.sciencedirect.com/. Accessed: 2019-08-16.

[12] R. Conejo, E. Guzmán, J. L. Perez-De-La-Cruz, and B. Barros. An empirical study on the quantitative notion of task difficulty. *Expert Systems with Applications*, 41(2), 2014.

[13] R. Conejo, B. Barros, and M. F. Bertoa. Automated assessment of complex programming tasks using siette. In *Proceedings of the IEEE Transactions on Learning Technologies*. IEEE, 2018.

[14] E. Gálvez, J.and Guzmán and R. Conejo. A blended e-learning experience in a course of object oriented programming fundamentals. *Knowledge-Based Systems*, 22(4), 2009.

[15] Eduardo Guzmán and Ricardo Conejo. Self-assessment in a feasible, adaptive web-based testing system. *IEEE Transactions on Education*, 48(4), 2005.

[16] E. Guzman, R. Conejo, and J. L. Perez-de-la Cruz. Improving student performance using self-assessment tests. *IEEE Intelligent Systems*, 22(4), 2007.

[17] H. Shimakawa, D. D. Phuong, Y. Yokota, and F. Harada. Refining programming education course with absolute measures. In *Proceedings of the International Conference on Education and Management Technology*. IEEE, 2010.

[18] G. R. Bergersen, D. I. Sjøberg, and T. Dybå. Construction and validation of an instrument for measuring programming skill. *IEEE Transactions on Software Engineering*, 40(12), 2014.

[19] S. Oeda and H. Kosaku. Development of a check sheet for code-review towards improvement of skill level of novice programmers. *Procedia Computer Science*, 126, 2018.

[20] P. Hubwieser, M. Berges, M. Striewe, and M. Goedicke. Towards competency based testing and feedback: Competency definition and measurement in the field of algorithms & data structures. In *Proceedings of the IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2017.

[21] D. Zingaro and L. Porter. Peer instruction in computing: The value of instructor intervention. *Computers & Education*, 71, 2014.

[22] S. Wang, Y. Han, W. Wu, and Z. Hu. Modeling student learning outcomes in studying programming language course. In *Proceedings of the Seventh International Conference on Information Science and Technology (ICIST)*. IEEE, 2017.

[23] L. Yan, A. Hu, and C. Piech. Pensieve: Feedback on coding process for novices. In *Proceedings of the ACM Technical Symposium on Computer Science Education*. ACM, 2019.

[24] P. Navrat and J. Tvarozek. Online programming exercises for summative assessment in university courses. In *Proceedings of the International Conference on Computer Systems and Technologies*. ACM, 2014.

[25] E. Wiebe, J. London, O. Aksit, B. W. Mott, K. E. Boyer, and J. C. Lester. Development of a lean computational thinking abilities assessment for middle grades students. In *Proceedings of the ACM Technical Symposium on Computer Science Education*. ACM, 2019.

[26] A. L. S. O. Araújo, J. S. Santos, W. L. Andrade, D. D. S. Guerrero, and V. Dagienė. Exploring computational thinking assessment in introductory programming courses. In *Proceedings of the IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017.

[27] T. Barker. An automated feedback system based on adaptive testing: Extending the model. *International Journal of Emerging Technologies in Learning (iJET)*, 5(2), 2010.

[28] O. Vozár and M. Bieliková. Adaptive test question selection for web-based educational system. In *Proceedings of the International Workshop on Semantic Media Adaptation and Personalization*. IEEE, 2008.

[29] M. Barla, M. Bieliková, A. B. Ezzeddinne, T. Kramár, M. Šimko, and O. Vozár. On the impact of adaptive test question selection for learning efficiency. *Computers & Education*, 55(2), 2010.

[30] Y. L. P. Vega, J. C. G. Bolaños, G. M. F. Nieto, and S. M. Baldiris. Application of item response theory (irt) for the generation of adaptive assessments in an introductory course on object-oriented programming. In *Proceedings of the IEEE Frontiers in Education Conference (FIE)*. IEEE, 2012.

[31] S. M. Čisar, D. Radosav, B. Markoski, R. Pinter, and P. Čisar. Computer adaptive testing for student's knowledge in c++ exam. In *Proceedings of the International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2010.

[32] S. M. Čisar, P. Čisar, and R. Pinter. Evaluation of knowledge in object oriented programming course with computer adaptive tests. *Computers & Education*, 92, 2016.

[33] M. Mohamad and A. J. Omar. Measuring cognitive performance on programming knowledge: Classical test theory versus item response theory. In *Proceedings of the World Engineering Education Forum (WEEF)*. IEEE, 2017.

[34] A. Aajli and K. Afdel. Conception and implementation of a computer adaptive assessment system for e-learning based on a new measuring skills approach. In *Proceedings of the International Conference on Intelligent Systems: Theories and Applications (SITA)*. IEEE, 2013.

[35] A. Aajli and K. Afdel. Towards a new approach of measure of skills applied to an adaptive assessments system used in e-learning and e-recruitment. In *Proceedings of the International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 2013.

[36] K. Mi Yang, R. J. Ross, and S. B. Kim. Constructing different learning paths through e-learning. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*. IEEE, 2005.

[37] A. E. Tew and B. Dorn. The case for validated tools in computer science education research. *Computer*, 46(9), 2013.

[38] A. E. Tew and M. Guzdial. The fcs1: A language independent assessment of cs1 knowledge. In *Proceedings of the ACM Technical Symposium on Computer science Education*. ACM, 2011.

[39] S. H. Edwards and K. P. Murali. Codeworkout: Short programming exercises with built-in data collection. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2017.

[40] M. Antal and S. Koncz. Student modeling for a web-based self-assessment system. *Expert Systems with Applications*, 38(6), 2011.

[41] G. Chen, J. Shen, L. Barth-Cohen, S. Jiang, and M. Huang, X.and Eltoukhy. Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 2017.

[42] A. Mühling, A. Ruf, and P. Hubwieser. Design and first results of a psychometric test for measuring basic programming abilities. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. IET, 2015.

[43] M. Berges and P. Hubwieser. Evaluation of source code with item response theory. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015.

[44] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the EASE*, 2008.