An experience in mixing cognitive and affective teaching approaches in a CS1 course

João Marcelo Borovina Josko[1]

[1]Federal University of ABC (UFABC)
Avenida dos Estados, 5001 – Santo André – SP – Brazil

marcelo.josko@ufabc.edu.br

Abstract. *Learning computer programming involves overcoming different obstacles to mature technical, cognitive, and social skills. The literature presents a variety of teaching approaches to engage students in learning how to program. However, there is a lack of works that combine different teaching methods from cognitive and affective dimensions or consider the latter dimension in face-to-face classes consistently. This work presents our experience mixing pair programming, formative feedback, aspects of the affective dimension, and creative programming problems. The preliminary results analysis of three groups (82 students) reveals the contribution of our approach to the pass and fail rates ($P = 0.0367$ and 0.0329, respectively) corroborated by students' feedback.*

1. Introduction

Gaining computer programming knowledge demands the incremental development of technical, cognitive, and social skills. It depends on the intrinsic (e.g., learning style, motivation, self-esteem, interest) and extrinsic factors (e.g., learning environment) of the students that challenge the teaching of introductory programming [Carbone et al. 2009].

The literature provides several educational approaches and information resources to support this endeavour. Numerous studies discussed teaching strategies based on collaborative learning [O'Donnell et al. 2015] or incidental education (e.g., game-based learning [Xinogalos et al. 2015]). Conversely, few works have presented pedagogical interventions that combine different cognitive methods, such as pair programming and jigsaw [Goel and Kathuria 2010]. Furthermore, we have not found works that interweave the teaching methods of cognitive and affective dimensions to address students differences (intrinsic characteristics and background) towards computer programming. The cognitive dimension focuses on concepts understanding, while the affective one concern students' emotions, feelings, and attitudes experienced on learning.

In this context, we report our experience using a novel approach (hereafter referred to as $MixOf4$) based on mixed teaching methods in an introductory programming course. This experience stresses two key contributions. Interconnecting pair programming, formative feedback, creative programming problems, and aspects of the affective dimension can provide a significant learning experience for students independent of their prior knowledge and emotional involvement with programming. Besides, it reminds the relevance of affectivity in computer science education.

This paper is structured as follows: In Section 2, we review the related works. Next, we characterize $MixOf4$ in Section 3, and we report its preliminary results in

Section 4. In Section 5, we discuss achievements and limitations, followed by some lessons learned in Section 6. We present conclusions and future works in Section 7.

2. Related Works

Literature provides several viewpoints to engage students in introductory computer programming learning. Due to space constraints, we considered in this section those viewpoints nearest to this work purposes.

The first viewpoint focuses on collaborative work to permit the social construction of programming knowledge. Such a collaboration process may be mediated by online collaboration tools [da Silva Estácio and Prikladnicki 2015] or conducted in traditional laboratory classes [O'Donnell et al. 2015]. The pair programming method has been broadly used in both settings and has rich support information and tips (e.g., guidelines). Some interventions have combined this method with the flipped classroom [Mok 2014], jigsaw-like approach [Goel and Kathuria 2010], media computation [Simon et al. 2010], gaming with programming microworlds [Xinogalos et al. 2015], and visual interactive learning [Iskrenovic-Momcilovic 2019].

Another viewpoint focuses on feedback-based approaches to promote learning on how to program. Some pedagogical interventions used formative feedback extensively with peer code review [Sun et al. 2019], scaffolding [Vikberg et al. 2013], or visual interactive learning and educational robots [Anfurrutia et al. 2018]. Others provided feedback through automatic assessment tools [Zampirolli et al. 2021]. Pair programming also uses feedback somehow, but it is not systematized and does not focus on formative information [Klopp et al. 2017]. However, a study by [Hahn et al. 2009] has shown the benefits of providing formative feedback to students after completing pair programming tasks.

The last viewpoint refers to the affective dimension as a factor of significant achievements in the teaching-learning process. The literature presents some studies that recognize the value of emotions and affective dimension in computer programming [Gurer and Tokumaci 2020] or computing science learning [Reynolds and Goda 2007]. Others concentrate on generating sensitive answers based on students' behavior interpretation through educational software (e.g., animated pedagogical agents [Johnson et al. 2016]). However, we have not found any work that systematically uses affective aspects in a face-to-face manner. Despite all previous contributions, no study has integrated pair programming, formative feedback, and affective elements to facilitate students' computer programming learning with different characteristics.

3. $MixOf4$ Description

At our university, the *introduction to programming* course aims at presenting programming fundamentals via problems covering the current curricula [Sahami et al. 2013]. This 12-week course is core for different academic units at our university, including Biomedical Engineering. After considering the course audience and students' anecdotal information, we realized that our teaching approach was inadequate.

Inspired by this self-reflection and the belief in the affective dimension's potential to leverage the learning process, we planned $MixOf4$[1], which involves

[1]For details regarding $MixOf4$ planning motivation and the selection and interaction of the methods, please visit here.

[Borovina Josko 2021]: encouraging collaborative work (Section 3.1), enriching this collaborative meaning construction by providing students with formative feedback (Section 3.2), strengthening professor-student bonds (Section 3.3), and assigning interesting programming problems that are assessed based on their functional coverage (Section 3.4). Due to our approach's nature, we adjusted the class planning (Section 3.5) and reinforced vocabulary consistency between all course materials.

3.1. Pair Programming

$MixOf4$ uses an ergonomic lab setup to permit students to pair comfortably and the professor to monitor students' interaction within each pair. This arrangement establishes a gap between each workstation to enable anyone to move freely. Moreover, our approach assumes that self-pairing leads to fewer compatibility problems [O'Donnell et al. 2015]. Hence, students are free to choose and keep their partners with just one exception; experienced computer programming students can not pair together.

Lastly, $MixOf4$ establishes that pair role transitions (*Driver* $\leftrightarrow$ *Navigator*) must occur around 50% of task completion or 50% of the lab class duration. This rule aims twofold: (i) ensure that pairs play both roles equally and (ii) avoid students to rely solely on their partners. During pair practices, the professor monitors to ensure pair programming protocol, provide feedback, maintain a positive environment, and stimulate team communication and collaborative culture.

3.2. Formative Feedback

$MixOf4$ uses feedback to help students understand their current weaknesses or strengths concerning each learning goal (e.g., using a matrix or nested loops). Its planning considers feedback elements (content, focus, delivery moment, exposition mode, and extension) effectively [Brookhart 2017]. Following this plan, we use two occasions to deliver feedback, mainly on programming tasks or pair-work skill. Personal feedback occurs orally on every student-professor contact during lab practices, scheduled meetings, or after lectures. At these moments, we coach students to improve their learning regarding a specific goal or invite them to think and discuss an issue (e.g., a particular problem solution, disagreement between pairs) without telling them what to do. This strategy is required to maintain students' sense of ownership.

The post-task feedback occurs up to 4 days after lab practices or individual exam completion. Based on students' assessment outcomes (Section 3.4), this textual feedback follows a format that highlights positive aspects and discusses improvement methods (e.g., additional exercises, videos) for each question topic. All feedback content uses a comprehensive technical vocabulary and friendly language (Section 3.3).

3.3. Affective Aspects

A pedagogical intervention includes several affective aspects contributing to the learning process. Recognizing the complexity of them, we adopted a careful and consistent approach. Therefore, $MixOf4$ considers two interrelated affective aspects of the professor's behavior to cultivate a learning environment on the grounds of a supportive learning environment [Hindman et al. 2013]. We are mainly concerned with promoting a supportive learning environment and alleviate emotional factors, including anxiety and fear.

The first aspect refers to the professor's approachability. It is concerned with fostering open, respectful, and friendly tutor-student and student-student relationship [Evans et al. 2009]. In every lecture (Section 3.5), we encourage students to share their doubts, provide a possible solution to a given problem, or contribute to explanations about a subject in their own words. We use some strategies (e.g., pair discussion, throw an object for the student to speak) to engage students in the classes. Furthermore, we clarify our extra-classroom availability in non-classroom situations to offer alternative explanations, discuss project advising, feedback information (Section 3.2), or other needs. For instance, we routinely take some time (usually five minutes) before and after lessons to chat with students.

The second aspect cares about the professor's style of communication. Our approach advocates promoting a climate of enthusiasm and a good mood to enliven his teaching [Hargreaves 1998]. For example, we usually discuss past programming situations funnily to make a topic clearer. Moreover, $MixOf4$ also assumes a spontaneous conversational style based on warm facial expressions and a tender voice in all teaching activities or personal interaction.

3.4. Assessment Characteristics

$MixOf4$ uses a set of summative (one individual project) and formative (nine lab practices) mandatory assessments solved out of the class and in lab time, respectively. This set provides hands-on experience in solving programming problems and allows an honest evaluation of students' individual and collaborative works. Each assessment comprises two to four mandatory questions that ask students to *solve* different programming problems. These problems are creative simplifications of situations inspired by data science (e.g., data manipulation), network traffic (e.g., compression), biology (e.g., DNA sequence comparison), and personal finance (e.g., investment decision), to name a few.

We adopt a gradual and decreasing method to grade students' solutions. This method follows conditioned two-step. In the first step, we test each solution according to a predefined set of test cases related to a given question. For those that do not reach 100%, we contrast (the second step) their answers with corresponding problem statements to determine students' functional coverage. Both outcomes are core to formulate the post-task feedback (Section 3.2) and grade students' solutions according to the reference criteria in Table 1.

Table 1. Partial Reference Criteria

Grade	Criteria Description
100%	All problem statements addressed. All case tests satisfied
90%	All problem statements addressed. Some case tests failed
30%	$\geq 30\%$ problem statements addressed. All corresponding case tests satisfied
20%	$\geq 30\%$ problem statements addressed. Some corresponding case tests failed

3.5. Classes Planning

Our introduction to programming is a 12-week course, as mentioned in Section 3. Each week comprises two lessons (2 hours long each) offered on different days of the week: a lecture followed by a laboratory practice. The first week is a special occasion where students taste our intervention based on $MixOf4$. In the first lecture, we explain the course characteristics and pair programming and contextualize the importance of algorithmic thinking and collaborative work. We also provide experimentation on the style of programming problems, pairing, and feedback in the first laboratory practice sample.

In the remaining weeks, we introduce programming topics (from basis to matrix) steadily and sequentially. Each lecture uses contextualized problems (e.g., home taxes) to allow the student to understand a given topic's properties. Students practice their understanding of the corresponding lab by solving programming problems (Section 3.4).

4. Preliminary Results

4.1. The Students' Profile

To gather student perceptions about the intervention based on $MixOf4$, we used two non-mandatory Google Forms-based surveys[2] (pre and post-course) with simple vocabulary. We applied $MixOf4$ in three groups (2018.1a, 2018.1b, 2019.1) of the introductory programming course to a total public of 82 (27 per group on average) enrolled students without disabilities. Of this amount, 78 students ($\simeq 95\%$) finished the course, and four ($\simeq 5\%$) have abandoned it. Of those that concluded it, 31 males and 19 females ($\simeq 60\%$ or $50/82$) filled out both surveys correctly. Their academic unit was Engineering ($\simeq 58\%$ or $29/50$), Computer Science ($\simeq 18\%$ or $9/50$), Chemistry (10% or $5/50$), Biology ($\simeq 6\%$ or $3/50$), Mathematics ($\simeq 6\%$ or $3/50$), or International Relations ($\simeq 2\%$ or $1/50$). The difference of final average grade between the students that concluded the course and that answered the surveys was negligible (7.2 and 7.5, respectively).

4.2. Performance and Dropout Rates

We identify exciting preliminary results of our $MixOf4$ based intervention from two different analysis on dropout and performance (pass and fail) rates. In the first analysis, we used descriptive statistics to contrast our groups' data (purple bars of Figure 1) with all institutional historical data (light blue bars of Figure 1). Due to many groups, we aggregated (by arithmetic mean) the older institutional groups' data in a range of time (first bar of Figure 1) and four-month period the remaining. In contrast, our groups' data we kept segregated.

The descriptive analysis of Figure 1 reveals that our intervention increases by 24% (from 63.5% to 79.3%) the mean pass rate. It also shows that our groups pass rate has a standard deviation ("Std Dev" solid lines) lower than the institutional (4.5% and 8.7%, respectively). Not shown in this figure, groups based on $MixOf4$ contribute to improving retention by decreasing the average dropout rate from 14.2% to 4.9% and reducing the average fail rate from 22.2% to 15.9%. It is worth noticing that some 2018.2 groups experienced an alternative schedule that makes classes hard to follow for some students.

Regarding the second analysis, we conducted a statistical significance test to evaluate the confidence of our groups' average performance and dropout rates. In other words, we tested the null hypothesis (H_o) that represents no change at all versus the alternative hypothesis (H_{alt}) that claims that our method does produce improvements. We met random (students chose the courses at random), normal and independence (our groups' size is less than 10% of $14K$ population) conditions for statistical inference.

[2]For supplementary surveys characteristics, please visit here.

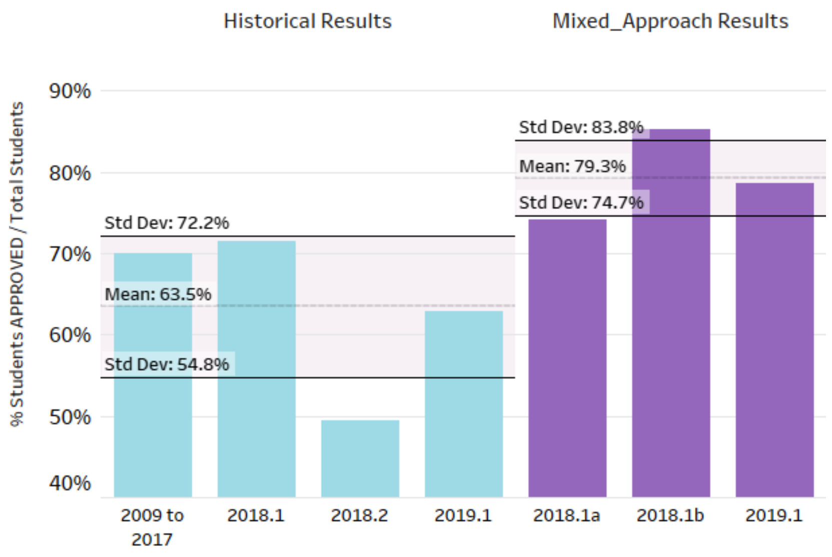

Figure 1. The average pass rate of groups that attended the classes using $MixOf4$ or not (Source: The author)

We calculated the z-scores, considering the entire students' population. After this procedure, we obtained $z_{pr} \simeq 1.79$, $z_{fr} \simeq -1.84$, and $z_{dr} \simeq -1.06$ for pass, fail, and dropout rates. These z-scores correspond to the p-values of 0.0367, 0.0329, and 0.1446, respectively. At a significance level of 95% ($\alpha = 0.05$), these values reject pass and fail rates null hypothesis and suggest that these results are significant.

4.3. The students view of $MixOf4$

We followed a three-cycle manual analysis process to scrutinize the open-ended questions of the post-survey. In the first cycle, we applied a descriptive coding method to establish the code list observed in Table 2. Following this base procedure, we performed a two-cycle investigation to highlight sentences or paragraphs according to the code list.

Table 2. Code list and their description

Code	Short Description
$R1$	Perception of overcoming and development
$R2$	Perception of barriers in the learning
$R3$	Perception of the usefulness/joy of the learning experience
$R4$	Perception of more confidence on programming
$R5$	Collaborative programming contribution to learning
$R6$	Affective aspects contribution to learning
$R7$	Task style contribution to learning
$R8$	Praises for the professor work
$R9$	Suggestions for course improvement
$R10$	Suggestions concerning a particular method

Figure 2 exposes 44 (84%) students answers regarding one or more aspects of $MixOf4$. Most students ($\simeq 66\%$ or $29/44$) express their feeling of overcoming and satisfaction with their learning process (code $R1$), especially those with no previous programming experiences. Quite a few students ($\simeq 11\%$ or $5/44$) declare difficulties in their learning process (code $R2$), especially related to the problems' complexity level. In contrast, most of them report the visibility of the knowledge acquired or improved, such as female student below.

"$S1$: At the beginning of the course, I was a little lost in what exactly programming was. I was also afraid and insecure about how it is going to work. Now that it had finished, I understand the topic better, and I feel more confident about it."

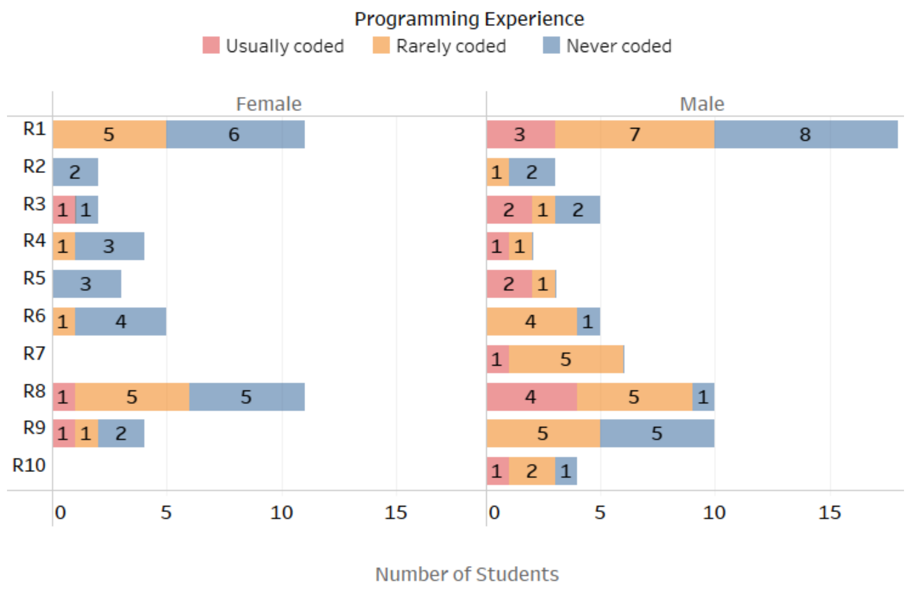

Figure 2. Students perceptions about their learning experience per code list and previous programming experience (Source: The author)

Concerning the methods used *per se*, students emphasize pair programming ($\simeq$ 13% or 6/44), style of programming problems ($\simeq$ 13% or 6/44), or affective aspects ($\simeq$ 23% or 10/44) contributions on their computer programming learning process (codes $R5$, $R7$, $R6$, respectively). The most-reported items refer to the possibility of discussing and evaluating different problem solutions with a colleague, the professor's affection and attention, and the engagement in solving relevant activities, corroborating [Goel and Kathuria 2010, Gurer and Tokumaci 2020, Mok 2014], respectively. It is interesting to note that female and male students with no or little programming experience prefer collaborative programming and professor attention. Below, we quoted female and male students' comments regarding these items, respectively.

$\mathcal{S}3$: "I have enjoyed very much doing the course with you. Your passion for teaching and attention toward us has motivated me to keep studying programming and get a passion for it."
$\mathcal{S}4$: "Teacher, your classes were not boring, because your exercises were really challenging and made us think of creative solutions. Thanks for the opportunity!"

Lastly, most of the suggestions focusing on the general aspects (code $R9$) of our approach ($\simeq$ 31% or 14/44) reveal the interest in game-based programming problems. This request (by male students) corroborates the way how students learn today, as acknowledged by [Xinogalos et al. 2015, Johnson et al. 2016]. Regarding a particular method (code $R10$), male students ($\simeq$ 9% or 4/44) recommend avoiding partner repetition in the lab classes and adopting peer assessment. The latter may suggest the students' perception for a more realistic assessment of pair tasks or because of problems related to pairing (as illustrated below).

$\mathcal{S}6$: "In my case, I felt a little harmed by some of my pairs, as it seems that they did not study the content before practical classes. So we wasted much time recalling concepts."

4.4. Programming Confidence Level

We used a paired t-test to compare students' programming confidence levels before and after our pedagogical intervention. In this work, the term "programming confidence" refers to how a student perceives his self-efficacy in solving programming problems. Assuming a significance level of 99% ($\alpha = 0.01$), we obtained a t-test of -12.691 (df=49, n=50) and a p-value of $2.2e-16$. This result rejects the null hypothesis that students' confidence remained the same.

Corroborating the mentioned result, Figure 3 shows the students' answers to the question "How confident do you feel programming a computer?" before and after attending our course. It exposes the sharp changing pattern of the students' programming confidence: from a negative perception to a more positive one. In quantitative terms, this pattern reveals that students' programming confidence increased from 16% ($(7+1)/50$) to 66% ($(15+18)/50$), while the uncertainty perception shrank almost 80% (from $(19+17)$ to $(1+7)$). Besides, the students' survey responses median shifted from 2 to 4.

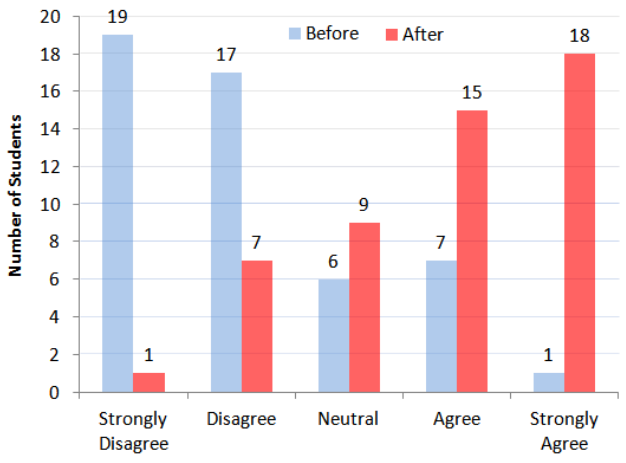

Figure 3. Students' programming confidence BEFORE and AFTER their learning experience (Source: The author)

5. Discussion and Limitations

This section reflects on achievements and limitations experienced during the pedagogical intervention based on $MixOf4$. Important to state that the professor that used such a novel intervention and the old one (Section 3) was the same. Considering the students' statements and performance results and our class-laboratory notes, $MixOf4$ provides four evident positive aspects.

The first refers to *joyfulness*. The vast majority of students show satisfaction and enthusiasm in pursuing solutions to the lab practices. This aspect links to the *first step towards a teamwork culture* (the second aspect). Several students reveal happiness to help or learn from a colleague (peer-learning), discussing different ideas until they reach a shared solution. A third aspect refers to *intense communication flow*. Most of the students participate actively and frequently in both lab practices and classes debates because of the "cool" learning environment (as some refer to it). Such communication flow reveals that students feel safe talking to their colleagues and the professor. The last aspect refers to *the amount of meaningful feedback* in professor-student interactions. Several students relate their level of achievement to this information.

However, $MixOf4$ has some limitations. Its characteristics make it fittable to a maximum of 25 students to ensure that the professor can adequately follow students' progress. Moreover, this approach is dependable on the affective and interpersonal traits of the professor.

6. Lessons Learned

We faced some challenges during the pedagogical intervention based on $MixOf4$. Due to space restrictions[3], we cite *handling different students' interaction styles*. Most students

[3]For further information regarding lessons learning and treats to validity, please visit here.

gradually developed real affective reciprocity with the professor while participating in lecture or lab classes. However, a few students showed no interest (or established limits) in a substantial relationship or dialogue with the professor. Hence, we employed a style (affectionate or reserved) according to students' inclination.

7. Conclusion

In this paper, we characterized our approach based on mixed teaching methods to deliver CS1 classes. Its preliminary results in three groups revealed that $MixOf4$ contributes to increasing students' success and programming confidence levels. Moreover, several students' feedback showed optimism about our approach, especially regarding the professor's approachability and the positive learning environment. However, many questions are still open.

In future works, we intend to conduct a longitudinal study with two aims: $i)$ investigate if groups based on our approach produce real higher levels of programming confidence than control groups and $ii)$ examine the interplay between the teaching methods used and how they support (or not) each other in helping students learning process according to their academic units.

References

Anfurrutia, F. I., Álvarez, A., Larrañaga, M., and López-Gil, J.-M. (2018). Integrating formative feedback in introductory programming modules. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 13(1):3–10.

Borovina Josko, J. M. (2021). Mixing cognitive and affective approaches in teaching introductory programming. In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 2*, pages 1–1, Germany. ACM New York, NY.

Brookhart, S. M. (2017). *How to give effective feedback to your students*. Association for Supervision and Curriculum Development.

Carbone, A., Hurst, J., Mitchell, I., and Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming. In *The Eleventh Australasian Conference on Computing Education*, volume 95, pages 25–34. Australian Computer Society, Inc.

da Silva Estácio, B. J. and Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. *Information and Software Technology*, 63:1–10.

Evans, I. M., Harvey, S. T., Buckley, L., and Yan, E. (2009). Differentiating classroom climate concepts: Academic, management, and emotional environments. *Kōtuitui: New Zealand Journal of Social Sciences Online*, 4(2):131–146.

Goel, S. and Kathuria, V. (2010). A novel approach for collaborative pair programming. *Journal of Information Technology Education: Research*, 9(1):183–196.

Gurer, M. D. and Tokumaci, S. (2020). Factors affecting engineering students' achievement in computer programming. *International Journal of Computer Science Education in Schools*, 3(4):23–34.

Hahn, J. H., Mentz, E., and Meyer, L. (2009). Assessment strategies for pair programming. *Journal of Information Technology Education: Research*, 8(1):273–284.

Hargreaves, A. (1998). The emotional practice of teaching. *Teaching and teacher education*.

Hindman, J., Grant, L., and Stronge, J. (2013). *The supportive learning environment: Effective teaching practices*. Routledge.

Iskrenovic-Momcilovic, O. (2019). Pair programming with scratch. *Education and Information Technologies*, 24(5):2943–2952.

Johnson, C., McGill, M., Bouchard, D., Bradshaw, M. K., Bucheli, V. A., Merkle, L. D., Scott, M. J., Sweedyk, Z., Velázquez-Iturbide, J. Á., Xiao, Z., et al. (2016). Game development for computer science education. In *Proceedings of the 2016 ITiCSE Working Group Reports*, pages 23–44.

Klopp, M., Gold-Veerkamp, C., Kuhn, M., and Abke, J. (2017). Can pair programming address multidimensional issues in higher education? In *International Conference on Interactive Collaborative Learning*, pages 479–486. Springer.

Mok, H. N. (2014). Teaching tip: The flipped classroom. *Journal of information systems education*, 25(1):7.

O'Donnell, C., Buckley, J., Mahdi, A., Nelson, J., and English, M. (2015). Evaluating pair-programming for non-computer science major students. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 569–574. ACM.

Reynolds, C. W. and Goda, B. S. (2007). The affective dimension of pervasive themes in the information technology curriculum. In *Proceedings of the 8th ACM SIGITE conference on Information technology education*, pages 13–20.

Sahami, M., Roach, S., Cuadros-Vargas, E., and LeBlanc, R. (2013). Acm/ieee-cs computer science curriculum 2013: reviewing the ironman report. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 13–14. ACM.

Simon, B., Kinnunen, P., Porter, L., and Zazkis, D. (2010). Experience report: Cs1 for majors with media computation. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 214–218.

Sun, Q., Wu, J., Rong, W., and Liu, W. (2019). Formative assessment of programming language learning based on peer code review: Implementation and experience report. *Tsinghua Science and Technology*, 24(4):423–434.

Vikberg, T., Vihavainen, A., Luukkainen, M., and Kurhila, J. (2013). Early start in software coaching. In *International Conference on Agile Software Development*, pages 16–30. Springer.

Xinogalos, S., Malliarakis, C., Tsompanoudi, D., and Satratzemi, M. (2015). Microworlds, games and collaboration: three effective approaches to support novices in learning programming. In *Proceedings of the 7th Balkan Conference on Informatics Conference*, pages 1–8.

Zampirolli, F. A., Borovina Josko, J. M., Venero, M. L., Kobayashi, G., Fraga, F. J., Goya, D., and Savegnago, H. R. (2021). An experience of automated assessment in a large-scale introduction programming course. *Computer Applications in Engineering Education*, pages 1–16.