Learning Styles Impact Students' Perceptions on Active Learning Methodologies: A Case Study on the Use of Live-coding and Short Programming Exercises

Andrés R. Masegosa, [1], Rafael Cabañas [2], Ana D. Maldonado [2] and María Morales[2,*]

[1] Department of Computer Science, Aalborg University, 2450 Copenhagen SV, Denmark; arma@cs.aau.dk
[2] Department of Mathematics, University of Almería, 04120 Almería, Spain; rcabanas@ual.es (R.C); ana.d.maldonado@ual.es (A.D.M); maria.morales@ual.es (M.M)
* Correspondence: maria.morales@ual.es

Abstract: This research paper explores the effectiveness of live-coding as an active learning methodology in teaching programming, particularly in the context of diverse learning styles. Live-coding, characterized by real-time coding demonstrations by instructors, has been increasingly adopted to enhance the learning experience in programming education. It offers immediate feedback, demonstrates problem-solving in action, and allows instructors to incorporate student suggestions, making it a dynamic and engaging teaching tool. However, its effectiveness varies among students with different learning preferences. This study investigates the impact of various learning style dimensions, as defined by the Felder-Silverman model, on the effectiveness of live-coding in an introductory object-oriented programming course. The study was conducted at Aalborg University, Denmark, with students from the BSc Software program. It aims to provide empirical evidence on how different learning style dimensions influence student preferences and effectiveness of live-coding, offering insights to educators for tailoring active learning methodologies in programming courses to diverse learner needs.

Keywords: Active Learning Methodologies; Live-Coding; Programming Education; Learning Styles; Felder-Silverman Model; Empirical Research

1. Introduction

Teaching a programming language poses unique challenges as it requires grasping complex concepts and developing practical skills [1]. Traditional teaching methods often struggle to engage students effectively in this domain. However, active learning methodologies, such as live-coding [2], have emerged as valuable tools to address these difficulties and enhance the learning experience for aspiring programmers [3]. Most of the experiences in bibliography report that observing the process of program planning and implementation makes a difference in student learning [4–6] and engagement [6–8]. In [9] an exhaustive study of published experiences with live-coding can be found.

Even though some studies [10,11] fail to find a significant difference between live-coding versus traditional approaches in students' performance in paper based exams, they find improvements such as a decrease in the extraneous cognitive load on the students thanks to live programming slows the pace of lectures [7,11]. This slower speed in explanations also allows the teacher to adjust the difficulty of the examples or to incorporate suggestions made by the students [8], centering the learning in the students [8] and increasing the students' engagement. On the other hand, this slowing down can complicate the coverage of learning goals [6], in consequence, fitting the programming speed so that students do not loose track while the objectives of the course are cover is one of the main challenges of live-coding [7,12]. Lin, Yeh and Tan [5] propose to record live-coding videos so that students could watch them out of class at their own learning pace. Using a blend methodology might also solve the difficulty reported by students [7] in staying focused when code blocks are large or during too long periods of live programming.

One of the main advantages of live-coding is the immediate feedback it provides [13]. Students witness code being written, executed, and debugged in real-time, promotes an iterative learning process. In this way, Fagernes and Gronli [7] suggest a repeated cycle of lecture, demo and exercise to benefit the learning process.

Another advantage is that live-coding makes mistakes become valuable learning opportunities [8] as students observe problem-solving techniques in action. By making mistakes, whether deliberately or inadvertently, helps the instructor [6,8] to show how to detect them as well as courses of actions or point out unorganized execution of programming tasks. Given that students value the opportunity to engage in the process of problem-solving [7], Berger [8] proposes the implementation of a mistake-oriented treasure hunt during live-coding sessions to capture students' focus and engagement in the learning activity. However, the process of rectifying defective code under time constraints and the attentive observation of numerous students introduces considerable complexity to an inherently cognitively demanding task [6,8].

But not only is this approach helpful when the code is shown by the instructor, Gaspar and Langevin [14] propose to switch the role of leading the live-coding from the lecturer to the students in a particular activity to improve the benefits of this methodology.

Nevertheless, live-coding has several drawbacks to consider. On the one hand, some environment factors should be taken account. For example, students could have problems to participate if the classroom lacks of computing supports, the projection screen does not let them see the code or the instructor does not properly control the area of the screen that is being projected, so the students have difficulties in identifying what they should be paying attention to [8]. Another challenge is monitoring students' understanding and performing in in-class exercises. In courses with a large number of students, some additional teaching support such as supplementary academic personnel [8] or peer-mentoring strategies [4] might be needed.

In addition, students have different learning styles because individuals have unique preferences and strengths when it comes to processing and retaining information [15]. These preferences and strengths can be influenced by various factors, including neurological, cognitive, and environmental factors. For example, individuals have distinct neurological structures, and brain processing capabilities can affect how they perceive, interpret, and process information [16]. Some students may have a greater affinity for visual stimuli, while others may excel in auditory processing or hands-on experiences [17]. Neurological variations contribute to the diverse learning styles observed in students [18].

The main assumption of this work is that different learning styles [17] can indeed influence the benefits of using live-coding in the classroom [2]. For example, for active learners, who exhibit a preference for interacting with learning materials through active participation in activities, the visual aspect of live-coding, where they can see the code being written and executed in real-time, seems to be highly beneficial. However, live-coding may not be as beneficial for those who prefer a solitary and reflective learning style. Thus, one could hypothesize that students who exhibit a strong preference for independent learning and introspection may find the interactive nature of live-coding less advantageous compared to other learning methods.

In this work, we aim to empirically investigate whether different learning style dimensions impact the students' preferences for how active learning methodologies, like live-coding and short programming exercises, are implemented in a programming course. This analysis is carried out in the context of an introductory course on object-oriented programming in the third semester of the BSc Software program at Aalborg University (Denmark). This work also aims to help guide other lecturers in the implementation of active learning methodologies in programming courses.

2. Theoretical Background

2.1. Live-Coding

Live-coding is a powerful and interactive teaching methodology that involves writing and demonstrating code in real-time in front of an audience, typically in a classroom or workshop setting [3,13]. It is particularly effective for teaching programming as an active learning methodology because it actively engages students in the learning process and provides them with a dynamic and immersive learning experience [10,19]. Here are some key aspects of using live-coding as an active learning methodology for teaching programming, according to the existing literature:

1. **Real-time demonstration**: Live-coding allows instructors to write and execute code in real-time, providing a step-by-step demonstration of the programming concepts and techniques being taught. This allows students to observe the thought process, problem-solving strategies, and debugging techniques employed by the instructor, enhancing their understanding of the programming concepts.

2. **Immediate feedback**: As the code is written and executed live, students can see the immediate results and outcomes of the code snippets being written. This provides instant feedback on the correctness and functionality of the code, helping students identify and correct errors or misunderstandings in real-time. Immediate feedback is essential for active learning and helps students build a deeper understanding of programming concepts.

3. **Interactivity and engagement**: Live-coding encourages active participation and engagement from students. They can ask questions, suggest modifications, and provide input during the coding session. This interactive environment fosters collaboration, critical thinking, and problem-solving skills. Students feel more involved and connected to the learning process, making it more engaging and enjoyable.

4. **Mistakes and debugging**: Live-coding allows instructors to demonstrate how to deal with mistakes and debugging, which are crucial aspects of programming. By witnessing the instructor's process of identifying and fixing errors in real-time, students gain valuable insights into effective debugging strategies, error handling, and problem-solving techniques. This hands-on experience prepares them for the challenges they will face when writing their own code.

5. **Visual and auditory learning**: Live-coding combines visual and auditory learning modalities, catering to different learning styles. Students can observe the code being written on a screen, read the code, and listen to the instructor's explanations simultaneously. This multimodal approach enhances comprehension and retention of the programming concepts being taught.

6. **Adaptability and flexibility**: Live-coding sessions can be adapted and adjusted based on students' progress and feedback. Instructors can modify the code examples on-the-fly to address specific questions or dive deeper into certain concepts. This flexibility allows for a personalized learning experience and ensures that the content is relevant and tailored to the students' needs.

Overall, live-coding as an active learning methodology for teaching programming offers a dynamic, engaging, and immersive learning experience. It encourages students to actively participate, experiment, and collaborate, fostering a deeper understanding of programming concepts and enhancing their problem-solving skills.

2.2. Felder-Silverman Learning Styles

The Felder-Silverman Learning Styles model (FSLSM) [17] was developed by Richard M. Felder and Linda K. Silverman in 1988 and is commonly referred to as the Felder-Silverman model or simply the Felder-Silverman Learning Styles. This model suggests that individuals have different patterns of preferences and tendencies in the way they learn and process information. The FSLSM proposes four factors or dimensions, with each dimension being an opposing pair of categories that represent an individual's preferred approach to learning. A person's inclination toward one category over another can vary in intensity, ranging from strong to moderate to mild. This preference strength, and possibly the preference itself, is not permanent but it can evolve with the person's education, life experiences, and the specific subject or context of instruction [20].

The four dimensions proposed by the FSLSM are:

1. **Information processing: Active or Reflective learners**. This dimension refers to how learners engage with new information. Active learners prefer to engage with the learning

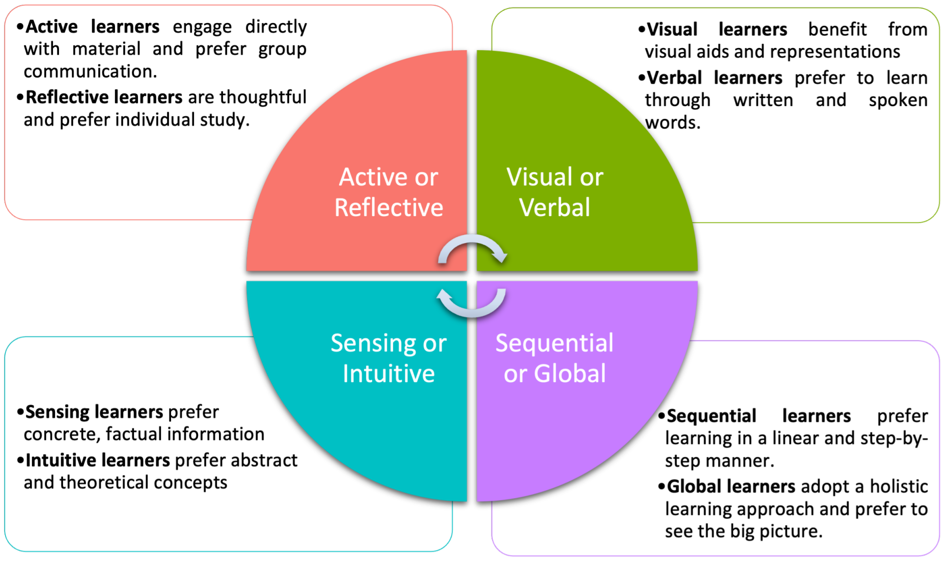

Figure 1. Felder-Silverman Learning Style dimensions

material through physical or interactive means. They learn best by doing, discussing, and actively participating in hands-on activities. They enjoy group work, experiments, and practical applications of knowledge. Active learners often thrive in collaborative environments where they can interact with their peers and instructors. Reflective learners, in contrast to active learners, prefer to process information internally. They are introspective and thoughtful, often taking their time to think through concepts before formulating their own understanding. Reflective learners benefit from quiet environments that allow for contemplation and introspection. They prefer individual study, writing, and self-reflection to solidify their understanding of new information, being fond of lectures and seminars.

2. **Information perception: Sensing or Intuitive learners**. This dimension relates to how learners perceive and process information. Sensing learners rely on their senses and prefer concrete, factual information. They appreciate practical examples, real-world applications, and hands-on experiences that allow them to engage with the subject matter in a tangible way. Sensing learners pay close attention to details, prefer step-by-step instructions, and may find memorization and repetition helpful in their learning process. Sensing learners tend to become distressed when faced with challenging tasks. They typically harbor concerns regarding the efficacy of academic practices and programs. Additionally, they dislike being assessed based on vague or embedded concepts and ideas. Intuitive learners are more interested in abstract and theoretical concepts. They enjoy exploring ideas, making connections between different concepts, and identifying underlying patterns and principles. Intuitive learners thrive in environments that encourage creativity, critical thinking, and conceptual understanding. They often seek out the bigger picture and are comfortable with ambiguity and uncertainty. Activities involving computation, rote learning, and conventional practices are unattractive to them.

3. **Information input: Visual or Verbal learners**. This dimension describes the preferred mode of input for learners. Visual learners learn best through visual aids and representations. They prefer information presented in the form of diagrams, charts, graphs, and visual illustrations. Visual learners exhibit a higher capacity for observation so they benefit from seeing relationships, spatial arrangements, and visual patterns. They may use color coding, mind maps, or visual organizers to enhance their understanding and retention of information. Verbal learners prefer to learn through written and spoken words. They excel in activities such as reading, writing, listening to lectures, and engaging in discussions. Verbal learners are skilled at understanding and remembering information when it is presented through words, explanations, and verbal instructions. They may benefit from reading textbooks, taking notes, and discussing concepts with others.

4. **Information understanding: Sequential or Global learners**. This dimension reflects the learners' approach to organizing and processing information. Sequential learners prefer learning in a linear and step-by-step manner. They thrive when presented with a logical progression of information and appreciate clear and structured instructions. Sequential learners prefer to understand each concept thoroughly before moving on to the next. They often excel in subjects that involve logical reasoning, problem-solving, and following well-defined processes. Global learners have a holistic approach to learning and prefer to see the big picture. They can quickly grasp overarching concepts and connections without necessarily needing detailed step-by-step instructions. Global learners enjoy synthesizing information, making associations, and understanding the broader context. They may struggle with tasks that require a linear approach or intricate, sequential details.

The FSLSM has garnered significant support for its comprehensive approach to understanding and enhancing educational experiences. Numerous researchers contend that learning styles play a crucial role in the educational process [21–24], underscoring the potential of the FSLSM to enhance instructional design and learner engagement by acknowledging and addressing diverse learning styles.Felder [20] points out the importance of the instructors being aware that each course contains students with different dimensions and that, to be effective, the instructor must adapt the methodology, activities and teaching material to match all students' dimensions. However, it is unfeasible to simultaneously align teaching with the learning styles of all students in a class. Whenever instruction aligns with one style, it inevitably diverges from one or more elements of the other styles present among the students. For this reason, it is generally recommended [25–27] to achieve balance across the dimensions, ensuring that students are exposed to both their preferred categories, enabling comfortable learning experiences, and their less favored categories, facilitating the development of crucial skills that may not be acquired through matched instruction alone. After reviewing 15 relevant publications, Sensuse et al. [28] propose five personalization approaches according the FSLSM by providing varied learning objects, diverse user interfaces, alternative learning paths, a variety of learning activities, and distinct representations of material content.

For identifying the learning styles of the students, Felder and Soloman [29] developed a questionnaire, called the Index of Learning Styles (ILS), that through 44 questions, attempts to identify the student's preference for each dimensions. Felder and Spurlin's validation study of the ILS [30] demonstrates the model's applicability and relevance in educational settings, offering a robust tool for assessing learning preferences. Furthermore, the psychometric study by Litzinger et al. [31] reinforces the reliability and construct validity of the ILS based on FSLSM, further solidifying its credibility and utility in educational research and practice. Although several studies find the ILS a valid and reliable instrument, others [32,33] point out some issues requiring further investigation. Moreover, the high number of items in the questionnaire has motivated some researchers to try to reduce the number of questions necessary to describe the preferences of the students. Graf et al. [34] examined the ILS to identify the five most characteristic questions for each dimension and, by using linear discriminant analyses to ascertain which questions were crucial in distinguishing between various learning styles, the authors reduce the questionnaire to 20 items. Haug et al [35] propose another short version of the ILS with 20 questions (five items per dimension) obtaining an accuracy rate higher than 86%. Finally, Goda et al. [36] develop another form with 12 items (three questions per dimension) and a correlation higher than 0.70 with the ILS classification.

But not only has the ILS received some criticism but also the FSLSM itself. Some research suggests that the effectiveness of matching instruction to specific learning styles is limited. The main criticism in these publications [15,33,37,38] is based on the lack of rigorous evidence of the so-called "meshing hypothesis", that is, tailoring instruction to suit students' learning styles optimizes learning outcomes. Indeed, in the bibliography we can find teaching experiences that report better results when the students' learning style is taken into account and experiences where no improvement occurs. Other critics [33,39] argue that the student's preferred way of learning is not necessarily the best way for them. In any case, there are no studies with an experimental design that provide sufficient scientific evidence for or against the improvement in learning or performance. The absence of supporting evidence is unquestionably crucial when considering the allocation of resources towards the implementation of a methodology founded upon the FSLSM framework, however, if the objective is to take into account the preferences of the students when receiving and processing information to create a learning environment more comfortable for students, these criticisms do not represent a real reason for not using learning styles, since a detrimental effect on learning has not been rigorously demonstrated either.

Regardless of the criticism, the FSLSM continues to be used in current research: El-Bishouty et al.[40] propose an interactive tool designed to enable educators in gauging the level of support a course offers for distinct learning styles, utilizing the FSLSM as its foundation. Nafea et al [27] present an advisor algorithm in machine learning that integrates students' actual ratings with their learning styles to provide personalized recommendations for course learning materials. In their study, the FSLSM is used to represent both the students' learning styles and the profiles of the learning objects. Other recent researches [28,41,42] recommend different personalization approaches based on FSLSM when developing a e-Learning system. Isal et al. [43] create and assess an adaptive mobile learning application designed to accommodate diverse student needs, with the FSLSM serving as the benchmark for identifying learning styles. Finally, the way to identify the learning style of students is the subject current research [44–46] , proposing new methods based on machine learning.

3. Methodology

The presented analysis is carried out on a specific group comprising Bachelor's degree students in Software Engineering at Aalborg University, Denmark. The participants were all in their third semester, aged between 20 and 22 years, and had prior knowledge of imperative programming. These students were enrolled in an Object-Oriented Programming course structured over 12 sessions, each lasting four hours. The course, taught using the Java programming language, aimed to enhance their understanding and skills in basic object-oriented programming concepts.

3.1. Lecturing and Short Exercises and live-coding

Throughout the semester, different active learning approaches were implemented to explore the level of engagement and comprehension, as well as to collect evidence about the preferences of the students.

In one setup, the lecturer focuses primarily on traditional lecturing. They deliver more in-depth presentations on key programming concepts. Visual aids such as slides, diagrams, and examples help illustrate the theoretical foundations of object-oriented programming. This setup allows students to gain a solid understanding of the fundamental principles before moving on to practical applications.

In another teaching setup, the lecturer introduces short exercises immediately after introducing a new concept, where students engage in hands-on practice through tasks designed to solidify their understanding and apply what they've learned. These exercises, which can be done individually or in pairs, facilitate active discussion and peer learning. Concurrently, the lecturer employs live-coding demonstrations to show the answer of the short programming exercises, providing real-time exposure to the object-oriented coding process. By coding from scratch and verbalizing their thought process, the lecturer demystifies problem-solving strategies and the rationale behind certain coding decisions. This method not only reinforces theoretical knowledge but also aids students in appreciating the practical application of object-oriented programming concepts, thereby fostering a more comprehensive grasp of the subject.

As the course progresses, the lecturer experiments with different combinations and proportions of lecturing and short exercises and live-coding demonstrations. For instance, they may reduce lecturing time while increasing the number and complexity of short exercises and live-coding demonstrations to provide more practical exposure.

3.2. Distribution of Time within a Four-Hour Session

The course is structured through a series of 12 sessions, each spanning a total of 4 hours. The sessions were divided into different formats to experiment with various ways of structuring the session.

2+2 Sessions: This is the standard format at Aalborg University. These sessions start with 2 hours mainly dedicated to lecturing, with some short exercises and live-coding, followed by an additional 2 hours for group exercises. This format allows for a more extensive exploration of the concepts covered in the first half of the session. It gives students the chance to engage in hands-on activities within a group setting, further reinforcing their understanding of object-oriented programming principles.

3+1 Sessions: In these sessions, the format combines traditional lecturing with many short exercises and live-coding demonstrations, during 3 hours. During this time, the instructor presents key concepts and demonstrates their practical application through coding examples. Students have the opportunity to actively participate in the learning process by completing short exercises and following along with the live-coding demonstrations. Additionally, these sessions allocate one extra hour for group exercises, encouraging collaboration and teamwork among the students.

4+0 Sessions: Lastly, this is a variation of the previous format, where the first part is extended to four hours. In this case, no time was allocated for group exercises.

4. Empirical Analysis

Considering the limited number of students and the concern that a survey with numerous items might deter participation, we chose not to employ the ILS recommended by Felder and Soloman [29]. This decision was based on the excessive length of the questionnaire in use. Furthermore, considering the benefit that could come from having students reflect on their own learning preferences [20], in the last session of the course a small seminar was organized based on the conclusions of the work carried out by Graf et al. [47]. In this seminar, the different dimensions of the FSLSM were explained to the students, using illustrative examples to highlight the differences between the different categories and opening a debate where students could clarify their doubts and share their impressions. After the debate, the lecturer conducted a digital survey to collect feedback about their preferences for the different teaching setups described in the previous section (session structures and the use of live-coding and short exercises) as well as to identify themselves according to the four Felder-Silverman learning style dimensions. A group of 32 students answered the survey. The survey can be found as supplementary material.

In the rest of this section, we show and analyze the survey data. Four main questions are addressed. For each of them, we show how the different learning style dimensions of the students directly impact the perceived benefits of the different teaching setups explored in the object-oriented programming course.

Moreover, we applied the Fisher's exact test to examine the significance of the association between the different learning style dimensions and the students' perceptions regarding 1) helpfulness of short exercises and live-coding demonstrations, 2) balance between theoretical and practical content, 3) attitude towards working alone or in groups, and 4) lecture setup.

4.1. Students' opinions about live-coding and short exercises

Students were asked about the perception they have regarding the use of short exercises and live-coding demonstrations interleaved during the lectures. As shown in Figure 2 (a), all of them thought they were helpful (38%) or very helpful (62%). We also examined if these answers were impacted by each learning style dimension. Figure 3 shows the distribution of these answers grouped by the different learning style dimensions. For example,

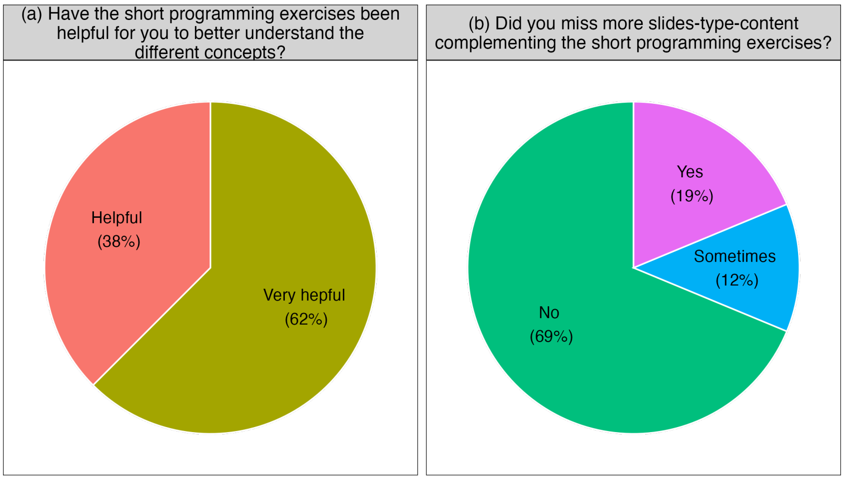

Figure 2. Students' perception of the helpfulness of the short exercises and live-coding demonstrations interleaved during the lectures (a). Students' answers regarding whether they missed more theoretical content supporting short exercises and live-coding demonstrations (b)

Figure 3 (a) displays how active vs. reflective learners perceived the benefits of live-coding and short exercises in the course. Students who identify themselves as active learners thought, in higher proportion, that live-coding and short exercises were very helpful, while around half of the students identified as reflective learners were not as positive.

According to this analysis, live-coding and short exercises seem to be especially helpful for active and sequential learners. However, at the 5% significance level, the statistical test applied revealed a lack of significant association between learning style dimensions and the students' perception in this case, as the minimum p-value observed was 0.4382. Although it is hard to elucidate the exact causes and the sample size of the survey is small, looking at the characteristics of each learning style, certain trends emerge and one could draw the following conclusions:

1. Active learners: Active learners thrive when they can engage in hands-on activities and participate actively in the learning process. Live-coding and short programming exercises provide them with opportunities to actively apply their knowledge, experiment with code, and see immediate results. The interactive nature of coding exercises aligns well with their preference for engaging with the material actively. Conversely, reflective learners often prefer to analyze and think deeply about information before forming conclusions. Live-coding and short exercises, which are more immediate and action-oriented, may not provide the level of reflective processing time that reflective learners typically need, which could have contributed to a lower perceived helpfulness among this group.

2. Visual/verbal learners: Visual and verbal learners show a lack of difference in their opinion about live-coding and short exercises. This could be due to the fact that these activities incorporate both visual and verbal elements, which could make these methods appealing to individuals with varying preferences on the verbal/visual dimension.

3. Sensing/intuitive learners: Sensing and intuitive learners showed almost the same perception towards the helpfulness of the live-coding and short exercises. These activities are considered active learning strategies that involve hands-on engagement. The fact that both sensing and intuitive learners showed similar opinions about these methods could indicate that active learning approaches may have broad applicability across the sensing/intuitive dimension.

4. Sequential learners: Sequential learners prefer learning in a step-by-step manner and appreciate a structured and ordered approach to information. Live-coding and short programming exercises often follow a sequential progression, starting with basic concepts and gradually building upon them. This sequential organization of programming exercises aligns well with the learning preferences of sequential learners. On the other hand, global learners tend to prefer a more holistic understanding of information. Live-coding and short

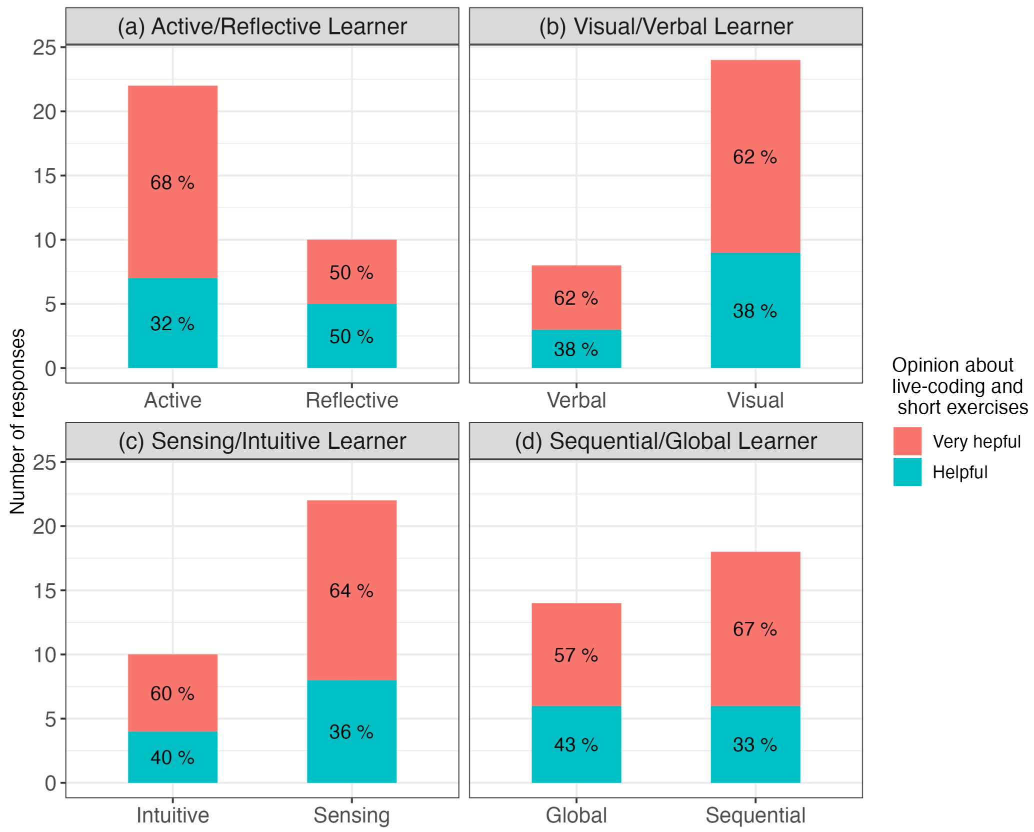

Figure 3. Students' opinions about live-coding and short exercises conditioned by learning style dimensions

exercises may not provide the level of holistic context that global learners typically seek, which could contribute to a lower perceived helpfulness among this group.

In summary, the combination of active engagement and sequential progression in live-coding and short programming exercises makes them particularly appealing to learners who possess these preferences.

4.2. Students' opinions about the balance between live-coding/short exercises and theoretical content

Although live-coding and short exercises are clearly helpful for students, *theoretical* content about main concepts of object-oriented programming, such as inheritance or encapsulation, was also provided in the course. In the survey, students were asked if the balance between live-coding-short exercises and *theoretical* content was adequate for them or if they missed the use of more *theoretical* content. As shown in Figure 2 (b), almost 70% of them did not miss more *theoretical* content (i.e., the balance was satisfactory), while the rest of the students missed it to varying degrees (answers "yes" and "sometimes"). As before, we also examined if these answers were impacted by each learning style dimension. Figure 4 shows the distribution of these answers across the different learning style dimensions.

According to this analysis, reflective, verbal, and intuitive learners missed, to a higher degree, the presence of more theoretical content about the main concepts of object-oriented programming. More precisely, 50% of reflective learners, 62% of verbal learners and 40% of intuitive learners missed theoretical content to some degree (answers "yes" and "sometimes"), as opposed to 23% of active learners, 21% of visual learners, and 27% of sensing learners. Global and sequential learners showed similar patterns, with 28% of global learners and 34% of sequential learners missing theoretical content to some degree. Despite the trends shown in Figure 4, the statistical test applied only showed significant association between verbal/visual learners and the students' preference regarding the

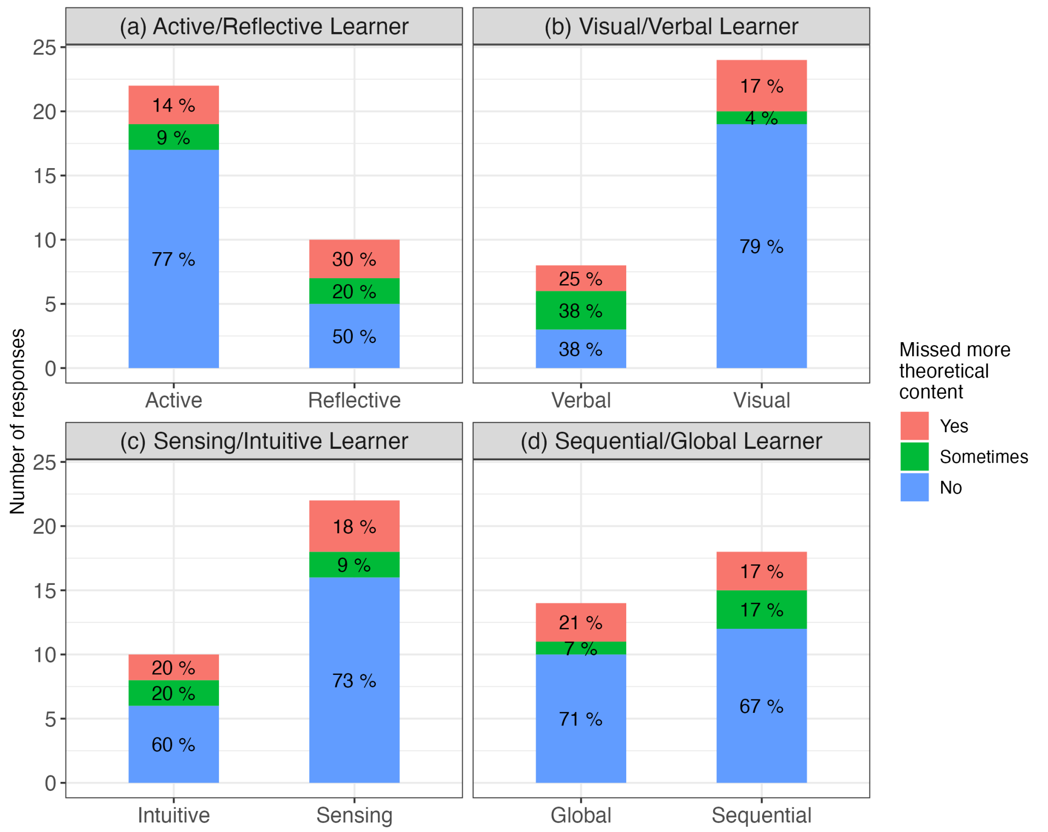

Figure 4. Students' opinions about balance between live-coding/short-exercises and theoretical content conditioned by learning style dimensions

balance (p-value = 0.0249), with visual learners being more likely to agree with the balance than verbal learners. Consistent with previous challenges in pointing out precise causes and constrained by a small survey sample size, an examination of the characteristics associated with each learning style allows for the formulation of the following conclusions:

1. Reflective learners: Reflective learners prefer to think deeply about information and internalize it before actively participating. They may have a preference for more theoretical content that allows them to analyze and reflect on the underlying principles of object-oriented programming. The absence of theoretical explanations may leave them feeling like they lack a solid foundation and a deeper understanding of the subject matter.

2. Verbal learners: Verbal learners learn best through written or spoken explanations and engage well with lectures and discussions. They may appreciate more theoretical content that provides detailed explanations of the main concepts of object-oriented programming. The absence of such content may hinder their ability to engage with the material and grasp the underlying theories behind object-oriented programming.

3. Intuitive learners: Intuitive learners have a natural inclination towards seeking patterns, connections, and understanding abstract concepts. While intuitive learners can often grasp programming concepts through practical application and hands-on experiences, they may still benefit from more theoretical content that provides a conceptual framework for object-oriented programming. The absence of theoretical explanations may limit their ability to make connections and understand the broader theoretical underpinnings of the concepts.

In short, while these learners may still benefit from practical exercises and hands-on learning, the inclusion of more theoretical content can cater to their preferences for deeper analysis, verbal explanations, and conceptual understanding.

4.3. Students' preferences for doing short exercises alone or in groups

In the third question, we asked students about their preferences for doing the short exercises, which were interleaved during the lecture, alone or in small groups. As shown in Figure 5, 72% of the students preferred to do the short exercises alone. In view of these data, it could be argued that students may prefer to do short programming exercises alone due to the advantages it offers. Working independently would allow them to concentrate and focus without distractions, leading to improved problem-solving abilities. It would also foster a sense of autonomy and independence, allowing students to approach coding tasks in their own way and take ownership of their programming learning process.

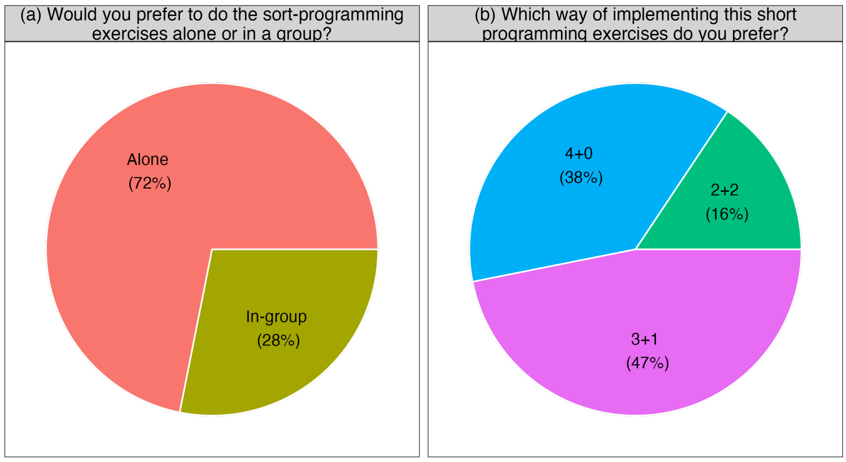

Figure 5. Students' preferences about doing coding exercises alone or in-groups (a). Students' preferences about the distribution of time within a four hours session (b)

Figure 6 shows the distribution of these answers across the different learning style dimensions. According to this analysis, reflective, verbal, sensing, and sequential learners preferred to do the short programming exercises alone rather than in small groups. Despite the trends shown in Figure 6, the statistical test applied only revealed a significant association between active/reflective learners and students' attitude toward working in groups or alone (p-value = 0.0303). Reflective learners were found to be more inclined to prefer working alone than active learners. Once again, articulating the precise causes remains challenging, complicated by the limited sample size of the survey. However, by examine the distinctive characteristics of each learning style dimension, it could be inferred the following conclusions:

1. Reflective learners: Reflective learners tend to prefer thinking and processing information internally before expressing their thoughts. They may find solitude conducive to deep thinking and reflection, allowing them to analyze and internalize the concepts at their own pace. Working alone on short programming exercises gives reflective learners the opportunity to focus their attention, contemplate their approaches, and thoroughly understand the code before seeking external input or engaging in discussions.

2. Verbal learners: Verbal learners often prefer written and spoken explanations and may benefit from articulating their thoughts and ideas through verbal communication. When working alone on short programming exercises, verbal learners have the chance to talk to themselves, explain their code aloud, or engage in self-directed discussions. This verbalization process can help them clarify their thinking, reinforce their understanding, and identify any gaps or areas that need improvement.

3. Sensing learners: Sensing learners typically prefer concrete, practical, and factual information. They may prefer to work individually on short programming exercises to focus on the specific details and practical application of code. Working alone allows them to concentrate on the specific syntax, algorithms, and implementation details without the potential distractions or variations that can arise in group settings.

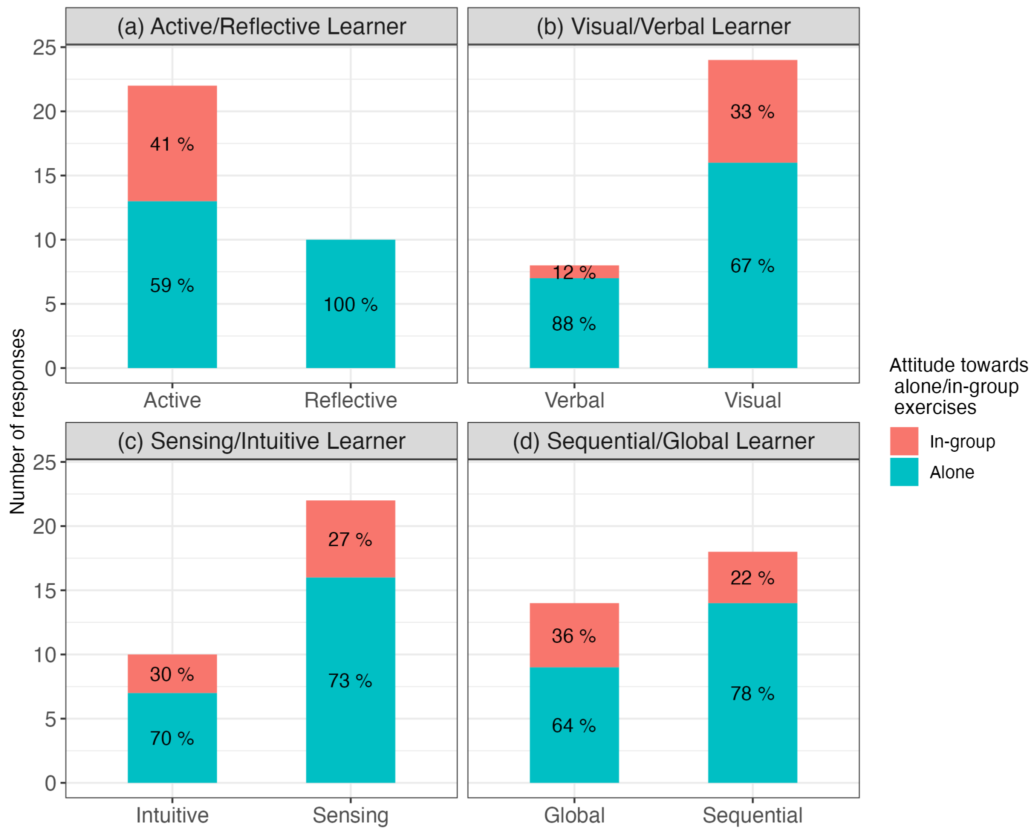

Figure 6. Students' preferences of doing short-exercises alone or in-groups, by learning style dimensions

4. Sequential learners: Sequential learners prefer learning in a linear, step-by-step manner, building knowledge piece by piece. They tend to appreciate structured approaches and organized thinking. When working alone on short programming exercises, sequential learners can follow a systematic process, carefully plan their steps, and work through the code methodically. They can progress through the exercise at their own pace, ensuring a logical sequence of actions and a thorough understanding of each step before moving forward.

In short, by working alone on short programming exercises, reflective, verbal, sensing, and sequential learners can align their preferred learning styles with their study habits. They can engage in introspection, verbalize their thoughts, focus on practical details, and follow a step-by-step approach, ultimately fostering a deeper understanding of programming concepts.

4.4. Students' preferences for the distribution of time within a four-hour session

As discussed in Section 3.2, each session spans a total of 4 hours, and different formats were used to structure the time of the session. Figure 5 (b) shows that only 16% of the students preferred the standard structure of a session at Aalborg University (2+2 sessions), while 47% of the students, preferred the 3+1 sessions.

In this case, we also analyzed how the different learning style dimensions may affect the distribution of the answers, but no significant differences in preferences were found at the 5% significance level, according to the statistical test applied. Figure 7 shows the distribution of these answers across the different learning style dimensions.

The standard session structure at Aalborg University (2+2 sessions), comprising 2 hours of lecturing and 2 hours of group exercises, emerged as the less preferred option among the three possibilities in all learning style dimensions. In contrast, sessions integrating many short exercises and live-coding demonstrations with traditional lecturing were generally welcome.

Within this approach, distinctions in preferences became evident between the 3+1 sessions (3 hours of lecturing/live-coding/short exercises + 1 hour of group exercises) and the 4+0 sessions (4 hours of lecturing/live-coding/short exercises, excluding group exercises). The 4+0 session was the preferred option among reflective (60%), sensing (45%), and sequential (44%) learners. Conversely, the 3+1 session was widely accepted among active (55%), intuitive (70%), and global (57%) learners. Both visual and verbal learners leaned towards the 3+1 session, with 46% and 50% of responses, respectively.

These results are consistent with the conclusions drawn regarding the students' attitude towards working alone or in groups. The 4+0 setup is preferred among reflective, sensing and sequential students probably because of the absence of group exercises at the end of the lecture.

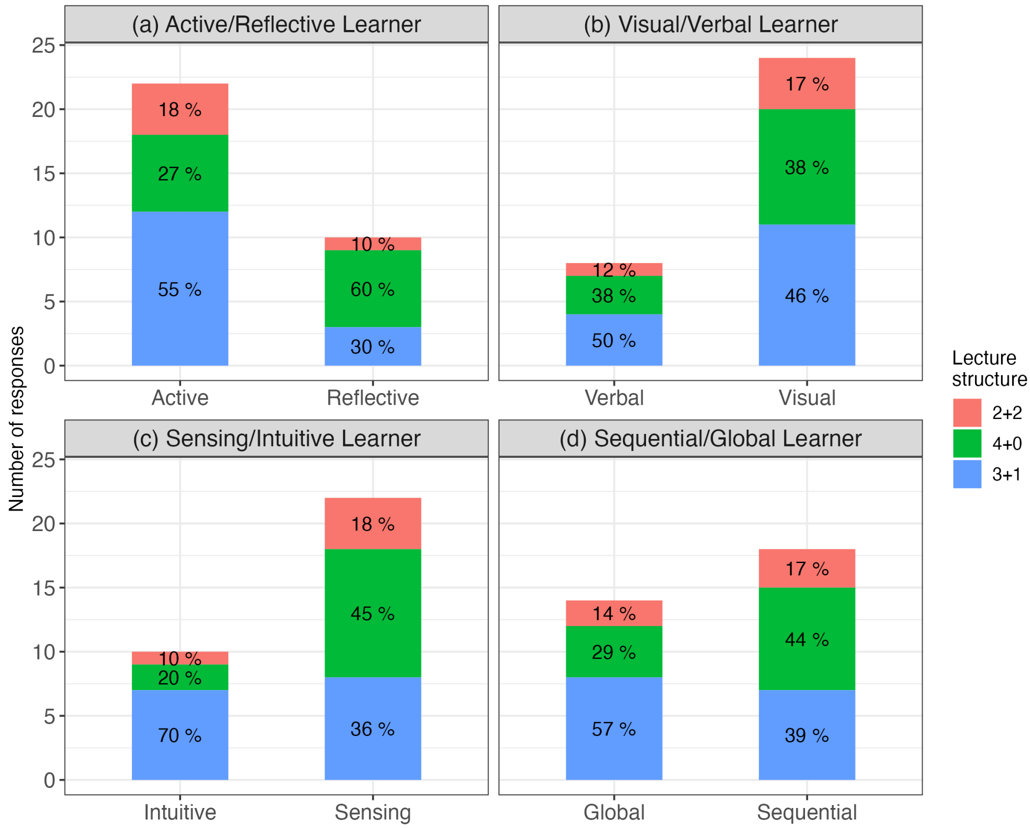

Figure 7. Students' preferences about the distribution of time within a four-hour session conditioned by learning style dimensions

5. Conclusions and Future Work

In conclusion, this study contributes to the existing literature by reaffirming that learning style indeed have an impact on the implementation of pedagogical methodologies. Consistent with previous findings, our empirical evidence demonstrates that learning style dimensions influence students' preferences regarding the implementation of active learning methodologies such as live-coding and short programming exercises. The observed patterns in the data tend to align with the abstract descriptions typically associated with each type of learner, as outlined in established learning style frameworks given by the Felder-Silverman model.

Furthermore, the study demonstrates how the utilization of live-coding aptly accommodates all dimensions of learning style, making it a methodology to consider when seeking to achieve a balance among the various categories within each dimension in course design.

The implications of this research could be valuable in the context of designing the course content for teaching a programming framework or a programming language. By taking into account the diverse learning style preferences of students, instructors can tailor their number of exercises and the use of live-coding to accommodate these differences.

References

1. Rodrigues, G.; Monteiro, A.F.; Osório, A. Introductory programming in higher education: A systematic literature review. In Proceedings of the Third International Computer Programming Education Conference, ICPEC 2022, June 2-3, 2022, Polytechnic Institute of Cávado and Ave (IPCA), Barcelos, Portugal; Simões, A.; Silva, J.C., Eds. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, Vol. 102, *OASIcs*, pp. 4:1–4:17. https://doi.org/10.4230/OASIcs.ICPEC.2022.4.
2. Johnston, L.W.; Bonsma-Fisher, M.; Ostblom, J.; Hasan, A.R.; Santangelo, J.S.; Coome, L.; Tran, L.; de Andrade, E.S.; Mahallati, S. A graduate student-led participatory live-coding quantitative methods course in R: Experiences on initiating, developing, and teaching. *Journal of Open Source Education* **2019**, *2*, 49.
3. Raj, A.G.S.; Patel, J.M.; Halverson, R.; Halverson, E.R. Role of live-coding in learning introductory programming. In Proceedings of the Proceedings of the 18th koli calling international conference on computing education research, 2018, pp. 1–8.
4. Chattopadhyay, A.; Ryan, D.; Pockrandt, J. Scaffolded Live Coding: A Hybrid Pedagogical Approach for Enhanced Teaching of Coding Skills. In Proceedings of the 2022 IEEE Frontiers in Education Conference (FIE). IEEE, 2022, pp. 1–9.
5. Lin, Y.T.; Yeh, M.K.C.; Tan, S.R. Teaching programming by revealing thinking process: Watching experts' live coding videos with reflection annotations. *IEEE Transactions on Education* **2022**, *65*, 617–627.
6. Schlichtkrull, A. An Experience with and Reflections on Live Coding with Active Learning. In Proceedings of the 4th International Computer Programming Education Conference (ICPEC 2023). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.
7. Grønli, T.M.; Fagernes, S. The live programming lecturing technique: A study of the student experience in introductory and advanced programming courses. In Proceedings of the Norsk IKT-konferanse for forskning og utdanning, 2020, number 4.
8. Berger, C.P. "I Feel Like I'm Teaching in a Gladiator Ring": Barriers and Benefits of Live Coding. PhD thesis, University of Maryland, College Park, 2023.
9. Selvaraj, A.; Zhang, E.; Porter, L.; Soosai Raj, A.G. Live coding: A review of the literature. In Proceedings of the Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, 2021, pp. 164–170.
10. Rubin, M.J. The effectiveness of live-coding to teach introductory programming. In Proceedings of the Proceeding of the 44th ACM technical symposium on Computer science education, 2013, pp. 651–656.
11. Raj, A.G.S.; Gu, P.; Zhang, E.; Williams, J.; Halverson, R.; Patel, J.M. Live-coding vs static code examples: which is better with respect to student learning and cognitive load? In Proceedings of the Proceedings of the Twenty-Second Australasian Computing Education Conference, 2020, pp. 152–159.
12. Nederbragt, A.; Harris, R.M.; Hill, A.P.; Wilson, G. Ten quick tips for teaching with participatory live coding. *PLOS Computational Biology* **2020**, *16*, 1–7. https://doi.org/10.1371/journal.pcbi.1008090.
13. de Castro, A.V.; Faria, L.; Cardoso, M.; Barroso, R.; Pereira, J.; Morgado, N. The potencial of real-time educational activities. In Proceedings of the EDULEARN18 Proceedings. IATED, 2018, pp. 4187–4197.
14. Gaspar, A.; Langevin, S. Active learning in introductory programming courses through student-led "live coding" and test-driven pair programming. In Proceedings of the International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL, 2007.
15. Pashler, H.; McDaniel, M.; Rohrer, D.; Bjork, R. Learning styles: Concepts and evidence. *Psychological science in the public interest* **2008**, *9*, 105–119.
16. Pulvermüller, F. How neurons make meaning: brain mechanisms for embodied and abstract-symbolic semantics. *Trends in cognitive sciences* **2013**, *17*, 458–470.
17. Felder, R.M. Learning and teaching styles in engineering education **2002**.
18. Newton, P.M.; Salvi, A. How common is belief in the learning styles neuromyth, and does it matter? A pragmatic systematic review. In Proceedings of the Frontiers in Education. Frontiers, 2020, p. 270.
19. McLean, A. Making programming languages to dance to: live coding with tidal. In Proceedings of the Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design, 2014, pp. 63–70.
20. Felder, R.M. Opinion: Uses, misuses, and validity of learning styles. *Advances in Engineering Education* **2020**, *8*, 1–16.
21. Collinson, E. A survey of elementary students' learning style preferences and academic success. *Contemporary Education* **2000**, *71*, 42.
22. Felder, R.; Henriques, E. Learning and Teaching Styles in Foreign and Second Language Education. *Foreign Language Annals* **1995**, *28*, 21–31.
23. Fida, A.; Ghaffar, A. Learning Styles: An Overview of the Felder-Silverman's Model and Measure. *International Journal of Innovation in Teaching and Learning (IJITL)* **2015**, *1*.
24. Tanner, K.; Allen, D. Approaches to biology teaching and learning: learning styles and the problem of instructional selection—engaging all students in science courses. *Cell biology education* **2004**, *3*, 197–201.

25. Felder, R.M.; Silverman, L.K.; et al. Learning and teaching styles in engineering education. *Engineering education* **1988**, *78*, 674–681.
26. Felder, R.M.; Brent, R. *Teaching and learning STEM: A practical guide*; John Wiley & Sons, 2024.
27. Nafea, S.M.; Siewe, F.; He, Y. On recommendation of learning objects using felder-silverman learning style model. *IEEE Access* **2019**, *7*, 163034–163048.
28. Sensuse, D.I.; Hasani, L.M.; Bagustari, B. Personalization strategies based on Felder-Silverman learning styles and its impact on learning: A literature review. In Proceedings of the 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE). IEEE, 2020, pp. 293–298.
29. Soloman, B.A.; Felder, R.M. Index of learning styles questionnaire. *NC State University. Available online at: http://www. engr. ncsu. edu/learningstyles/ilsweb. html (last visited on 14.05. 2010)* **2005**, *70*.
30. Felder, R.; Spurlin, J. Applications, Reliability, and Validity of the Index of Learning Styles. *International Journal of Engineering Education* **2005**, *21*, 103–112.
31. Litzinger, T.; Lee, S.; Wise, J.; Felder, R. A Psychometric Study of the Index of Learning Styles. *Journal of Engineering Education* **2007**, *96*, 309–319.
32. Viola, S.R.; Graf, S.; Leo, T.; et al. Analysis of felder-silverman index of learning styles by a data-driven statistical approach. In Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06). IEEE, 2006, pp. 959–964.
33. Kirschner, P.A. Stop propagating the learning styles myth. *Computers & Education* **2017**, *106*, 166–171.
34. Graf, S.; Viola, S.R.; Leo, T.; Kinshuk. In-depth analysis of the Felder-Silverman learning style dimensions. *Journal of Research on Technology in Education* **2007**, *40*, 79–93.
35. Haug, J.; Fischer, D.; Hagel, G. Development of a Short Form of the Index of Learning Styles for the Use in Adaptive Learning Systems. In Proceedings of the Proceedings of the 5th European Conference on Software Engineering Education, 2023, pp. 194–198.
36. Goda, Y.; Arame, M.; Handa, J.; Toda, M.; Matsuba, R.; Zhou, H.; Itoh, M.; Kitazaki, S. Development of a Short-Form Learning Style Inventory for Automated Driving Safety Education. In Proceedings of the 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), 2020, pp. 847–851. https://doi.org/10.1109/TALE48869.2020.9368318.
37. Willingham, D.T.; Hughes, E.M.; Dobolyi, D.G. The scientific status of learning styles theories. *Teaching of Psychology* **2015**, *42*, 266–271.
38. Newton, P.M.; Miah, M. Evidence-based higher education–is the learning styles 'myth'important? *Frontiers in psychology* **2017**, *8*, 444.
39. Knoll, A.R.; Otani, H.; Skeel, R.L.; Van Horn, K.R. Learning style, judgements of learning, and learning of verbal and visual information. *British journal of psychology* **2017**, *108*, 544–563.
40. El-Bishouty, M.M.; Aldraiweesh, A.; Alturki, U.; Tortorella, R.; Yang, J.; Chang, T.W.; Graf, S.; et al. Use of Felder and Silverman learning style model for online course design. *Educational Technology Research and Development* **2019**, *67*, 161–177.
41. Sihombing, J.H.; Laksitowening, K.A.; Darwiyanto, E. Personalized e-learning content based on felder-silverman learning style model. In Proceedings of the 2020 8th International conference on information and communication technology (ICoICT). IEEE, 2020, pp. 1–6.
42. Joseph, L.; Abraham, S.; Mani, B.P.; N, R. Exploring the effectiveness of learning path recommendation based on Felder-Silverman learning style model: A learning analytics intervention approach. *Journal of Educational Computing Research* **2022**, *60*, 1464–1489.
43. Isal, R.; Santoso, H.; Novandi, E. Development and evaluation of a mobile-learning application based on the felder-silverman learning styles model. *International Journal of Emerging Technologies in Learning (iJET)* **2021**, *16*, 107–124.
44. Ikawati, Y.; Al Rasyid, M.U.H.; Winarno, I. Student behavior analysis to predict learning styles based felder silverman model using ensemble tree method. *EMITTER International Journal of Engineering Technology* **2021**, *9*, 92–106.
45. Valencia Usme, Y.P.; Normann, M.; Sapsai, I.; Abke, J.; Madsen, A.; Weidl, G. Learning Style Classification by Using Bayesian Networks Based on the Index of Learning Style. In Proceedings of the Proceedings of the 5th European Conference on Software Engineering Education, 2023, pp. 73–82.
46. Wanniarachchi, W.; Premadasa, H. Integrated Model for Identifying the Learning Style of the Students using Machine Learning Techniques: An Approach of Felder Silverman Learning Style Model20 **2023**.
47. Graf, S.; Viola, S.R.; Kinshuk, T.L. Representative characteristics of felder-silverman learning styles: An empirical model. In Proceedings of the Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2006), Barcelona, Spain, 2006, pp. 235–242.