View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by BIBSYS: Open Journals Systems

The Live Programming Lecturing Technique: A Study of the Student Experience in Introductory and Advanced Programming Courses

Siri Fagernes and Tor-Morten Grønli
Mobile Technology Lab, Department of Technology,
Kristiania University College
Oslo, Norway

Abstract

This paper investigates the topic of teaching programming in higher education. The teaching method often referred to as *live programming* has become a widely applied lecturing strategy for teaching programming subjects in an interactive fashion. Lectures based on live programming normally involve live demonstrations, explanations and interaction with the students. Although this technique seems to be very popular amongst students and instructors, we hypothesise that it also involves potential challenges. In this paper, we investigate the perceived difficulty and promise of following such an approach from a student perspective. We present results from interviews with 1st and 2nd year IT Bachelor students about their experience with live programming. Our results indicate that students' engagement and desire to learn through active learning techniques still are very much valid also in introductory and advanced programming courses. Furthermore, we also interpret from our findings a suggested model of a repeated cycle of lecture, demo and exercise as highly beneficial to the student learning process.

1 Introduction

Programming courses constitute an important integral part of most Bachelor programmes in informatics, information technology, computer science or related programmes in higher education throughout the world. Despite an increasing focus on exposing kids and young adults to programming at an early age through initiatives such as the Norwegian project *Kidsa Koder* [1], or the related international initiatives *Computing at School* [2] and *Code Club* [3], programming is still primarily taught in higher education. The nature of programming when coming from a general, non-information technology, background is the concept of the domain-specific approach and vocabulary, put to use through advanced tools and they must become proficient within a short timeframe [4].

A combination of student active learning strategies, different pedagogical models, and combined efforts of lecturers and student assistants, usually encompass the team aimed at

This paper was presented at the NIK-2020 conference; see http://www.nik.no/.

solving the task of educating the students. The majority of the students rarely have any large, structured knowledge base regarding programming and a minority of the cohorts are usually self-taught from Internet, YouTube and online courses such as Udemy[1], Udacity[2] or Lynda[3]. Similarly, a small portion of the students may have prior higher education or some industry experience which creates a small fundament. However, the lecturers task is to bring this majority of unexperienced, unexposed and heterogeneous group up to a level of basic understanding and mastering of the topic of programming.

A much widespread technique to achieve this goal for the lecturer is to use the concept of *live programming*. Live programming refers to the process of designing and implementing a coding project during a lecture [5]. This can be conducted in several ways. One technique is to create an example application from scratch during the lecture, often with frequent interaction with the students. Another common approach is to prepare the "skeleton" of an application in advance, and fill this template with content, either from copy-paste or from writing line by line throughout the lecture. A third approach is to use collaborative tools to activate the students in participating in developing an application, as showcased by Gonzalez and Lauvås [6].

Earlier studies such as [7, 8] investigated live programming used in introductory courses. However, their focus was primarily on the perspective of the lecturer, and *how* live programming can be used as a lecturing technique. We would like to shift the focus to the perspective of the student. Instead of measuring the learning outcome, we concentrate this research on their experience of this widespread technique and try to understand how it is valued by the students. Additionally, we bring a new perspective into this topic by not only investigating introductory programming courses. To the best of our knowledge, there is a lack of research investigating this phenomena in *non introductory* courses, and we seek to ask the same questions to students of an advanced programming course.

Our research question in this article is *from the student's perspective, what are the experienced positive and negative effects on their learning process from live programming lectures?*

The rest of the paper is structured as follows. Firstly, we present selected literature in related work, and then we describe our methodological approach. To follow, our results are presented and we discuss their implications. Finally, we conclude the paper and indicate directions for future research.

2 Related Work

As discussed early on by Bonwell et al. [9], and later in-depth investigated by Weltmann [10], active learning facilitates for larger involvement and increased activity from the students besides just listening to the lecturer. Reading, writing, discussing, solving problems, coding, tutorials, and participating in group exercises, are just a snippet of examples of this educational technique. Engagement in the class room and facilitation of active learning can very well be many-faceted and are broadly explored in literature, but still academics diverge rather than agree on a common definition. Following the thoughts of Prince [11] we find a core characteristic of active learning by seeing it as a meaningful, cognitive challenging task undertaken by students while a lecture is given - often in relation to a proposed task or question from the lecturer.

[1] `https://www.udemy.com`
[2] `https://www.udacity.com`
[3] `https://www.lynda.com`

Active learning is not a replacement or substitute for traditional lectures, but rather a philosophy of conscious choice of strategies for teaching a topic to a group of students. The effect of active learning was investigated by Freeman et al. [12] and their results indicated an improved student performance, unrelated of class size, course type or level, across *science, technology, engineering and mathematics (STEM disciplines)* which were investigated. Improving learning outcomes through engaging the students to be more active in the learning process is not new. The idea was already visited by Bloom [13] in his early works, then in relation to his taxonomy of learning objectives.

Studying active learning strategies in relation to introductory computer programming are reported from several studies. An interesting perspective is brought up by Seeling and Eickhold [14] in their research investigating three varieties of active learning in an introductory programming course during three semesters. One with traditional lectures, the second with active learning, and the third with workshop- or lab-based sessions. They experienced that a shift from traditional lecturing to active learning produced inconclusive results. Overall exam performance increased, overall course performance where sustained, and course evaluation of mastery declined. Investigating their data, they conclude that the active learning enhanced significantly results of lower performing students, but otherwise were insignificant when compared across all result categories. Further studies will be conducted in a mix between lab-session and active learning lectures. Interestingly, they do no investigate or ask the students, on how their experience on the subject matter is.

We do seem to find abundance of studies about different perspectives of live programming in introductory programming classes and their positive effect. One such study is from Rubin [8], which presented a study on evaluating the effectiveness of using live programming in programming classes. This research analysed the outcome based on two surveys, and the results indicate that students prefer live programming compared to only being shown static code examples.

Shannon and Summet [7] did a study investigating the impact of live programming in a student-active learning context. Their main motivation was to determine how effective live programming is for specific concepts within programming courses, like if-statements, loops, methods and arrays, and to explore whether this strategy has more impact for certain topics than others, and to see the short-term learning effect. Their concept of live programming referred to students working on programming exercises in class in a flipped-classroom context. To measure the effect of each session the authors used in-class quizzes during the course to determine the short-time learning effect. Based on their study, it is difficult to evaluate the effect of live programming, as this work is really about active learning techniques. Introducing the quizzes may also have had an effect on the learning outcome of the sessions.

Gaspar and Langevin [15] investigated the use of so-called student led live programming, showing the benefits of letting the students engage in active learning during the programming lectures. The authors emphasise the importance of students learning the process and strategies behind programming, and not just the syntax of a program. Their results showed that the student led approach, combined with assignments with more than one possible solution, was successful in the sense that the students were creative in applying a range of different techniques to solve the given problem.

3 Methodology

This paper builds upon the qualitative tradition, and data have been gathered through the use of semi-structured interviews [16]. An interview guide with open ended questions was developed, and through traversing back and forth in the prepared questions we investigated the conceptual phenomena. Semi-structured interviews were selected since we are studying concepts and trying to understand and identify relevant issues from the student perspective. A total of 10 participants were interviewed over a period of three weeks.

The Participants

The participants were recruited among students in the first and second year of the Bachelor in information technology at Westerdals Oslo ACT. The first year students were chosen on the basis that they are regularly exposed for live programming in their 7.5 ECTS course of *Object Oriented Programming 2*. Similarly, the second year students were recruited from a 7.5 ECTS course in *Android programming* that exposes them to live programming on a weekly basis in an advanced course. Table 1 gives an overview of the year of study distribution over gender and age-span.

Table 1: Participants

Participant	**Semester**	**Gender**	**Age**
P1	4	Male	36
P2	4	Male	24
P3	2	Male	26
P4	2	Female	25
P5	2	Female	22
P6	4	Male	28
P7	2	Female	24
P8	4	Male	25
P9	4	Male	27
P10	4	Male	26

The Interviews

All interviews lasted between 20 and 40 minutes and followed a semi-structured style. The participants were informed of the purpose of the study, were offered to receive the finished article when published, and were given the option to end the interview or withdraw at any time.

Our purpose with the interviews was to get an understanding of the students' individual learning experience related to the live programming in the lectures they had attended. We had prepared an interview guide of open-ended questions, and a conversational approach was followed in the interviews with the students. During the semi-structured interviews questions related to the following topics were asked:

- The amount of live programming used in the programming courses they had attended so far in their education, and in what manner the live programming had been carried out by the instructors.

- How the student chose to engage in the live programming sessions, for instance if they would try to code along with the instructor throughout the lecture, or if they would rather pay attention to the programming without coding themselves.
- Their personal opinion of how effective live programming is related to their learning, with a particular focus on what they regarded potential benefits using this teaching strategies.
- Any potential challenges or issues related to the live programming sessions.
- If the students had any suggestions to possible improvements that could be done to the way it has been carried out, based on their experience.

4 Results

Based on the survey as detailed in section 3, we will further present the results in relation to the topics we used as starting point for our interviews.

Live programming teaching styles

The student responses indicate that live programming is being taught using several different approaches. While some instructors edit code in prepared code snippets, others perform live programming of smaller and larger applications from scratch. In some cases, the live programming is done for shorter intervals during a lecture containing other lecture components, while others choose to perform the entire lecture in a code-along manner. The difference in teaching approaches may be related to pedagogical strategies of the lecturer and his or her individual preference. Some students commented on this as being an advantage as they were exposed for several different methods, whereas other found this a distracting factor.

The students reported that there is also a significant difference between lecturers with the respect to the programming speed during the live sessions, and this referred to both the coding and the switching between the files being edited in the lecture. It was also mentioned that there is a lot of variation in how much the instructor explains along the way. One student said that some instructors are good at explaining, while others that are very experienced themselves, seem to forget how many small details that need to be correct for the whole program to work properly.

Some students reported that they mostly have experienced that live programming involved small snippets of code, and when larger programs are displayed, it is normally pre-coded, and just modified during the live-session. Other students reported experiences of lecturers spending a large amount of time on live programming and to produce programs of larger size and increased complexity.

The students also explained how different kinds of tools are being used by different lecturers. The *Scrimba* tool [6], which allows for displaying code from the audience on the lecturer's screen, was mentioned as being very useful, although a bit scary at first.

Student activity during lectures

When asked about how they choose to participate during the live programming sessions, the students varied in their responses. Many of the students said that they would live code along with the lecturer in the beginning, when the code snippets were small and the programming speed sufficiently low for them to keep up. Some also reported that

they would often give up, and simple pay attention to what was going on when the programming speed increased, or that they got lost of other reasons (P7).

P9 described how he initially tried to code along with the lecturer, but how he ended up simply typing off the lecturer without also perceiving the explanations that were given along with the code bits. After a while the student chose to skip coding along, and instead paying attention to the live programming and the explanation. After the lecture, the student would code the examples from the session on his own.

Another student (P10) explained that he sometimes codes along with the lecturer, and other times not, and that this is strongly related to the student's energy level each particular day. If he is tired, he will normally just pay attention, and if the energy level is higher, he programs along with the instructor. The student specified how he felt the live coding is valuable independently of whether he codes along or not since he always has the opportunity to ask questions during the lecture and participate actively that way.

A couple of students (P3, P9) chose to be active through the live coding by taking notes throughout the session, and explained how they felt that they got a better overview of what is going on this way. P3 would sometimes write down some of the code by hand, to get a better understanding.

Benefits

All of the participants reported that they enjoy and prefer live programming as part of the programming lectures. P6 stated that he learns most from coding along with the lecturer, and tries to do a mix of coding along and watching what is being presented. The participants were also quite uniform in their answers with respect to what they like about the live programming. Several mentioned how they appreciate experiencing the process of problem solving, and one specified how this is particularly when it is done from scratch. Further, when comparing to simply being exposed to pre-coded programs on a slide, several students specified how they prefer seeing the program being built, and not only reading text, *"just seeing the final code does not give you any hint of how it was built, or the sequence of how things are done" (P9).*

Another element mentioned by several students was the perceived interactivity in the live programming sessions. Some referred to how it is easier to pay attention because something happens, and how the lecturer can respond to comments and questions, make changes to the code and explain better what is going on. It was also commented on how live programming slows the general lecture speed down, which makes it easier for them to follow and digest the material (P2). One also reported how he enjoys observing an experienced programmer, and how he learns the most from how they perform the entire process from beginning to end.

Challenges

With respect to what is perceived as drawbacks or challenges with live programming during lectures, most of the students mentioned issues related to the speed of programming. *"If it is done too fast, it is easy to lose track, and difficult to get back in. Looking down at the wrong time can make you fall out of the coding completely." (P2)* It was also mentioned how it can be perceived as difficult to ask questions if you are stuck. One of the first-year students (P7) mentioned how syntax errors are frequent and difficult to spot when you are inexperienced in programming, also adding to the difficulty of coding along with the lecturer.

While a slow pace was referred to as a good thing by many, one student (P2) stated that too long periods of live programming without any disruptions can be very dry or boring. In that context it was advised to use a blend of live coding and other lecturing elements during the session. P6 mentioned that it could be difficult to stay focused on live coding during lectures if the code blocks are too large. He also mentioned how sometimes the lecturers are not able to complete the programming in such cases.

Several students also mentioned how good explanations along the way are necessary. *"It is very difficult if one does not understand individual parts of the code, particularly if no explanations are given with respect to why the code is there and what it does. That can be very stressful."* It was also mentioned by students in the first year, that it is important to remember to explain elements that may be obvious to the more experienced programmers.

Context switches were also reported by P3 as a significant challenge, for instance when the lecturer moves between different windows and files during the programming. Also, it was stated that complex applications are not suited for live programming, as this makes the student dividing the attention between paying attention to understand and typing the code (P2).

Experiencing code that does not compile or work appropriately was also referred to as problematic and stressful. Some students also commented on how they relied on having up-to-date software and hardware to be able to keep up with the lecturer, due to technical problems with running the code otherwise. Some also mentioned how the instructor sometimes struggle with getting things to work properly, and how this could be very time-consuming.

How to improve the learning experience

When asked how the learning experience could be improved, several students named finding the appropriate programming speed to be the most important. Many of them also expressed a wish for better explanations, like choice of strategy along the way, and other issues related to the actual problem solving. A highlighted idea was to show the context of the live programming, for instance through a switch between diagrams or illustrations and the coding. By doing this, more explanations of trivial matters would be explained several times, and combined with lower speed, these detailed explanations were expected to improve the learning process. One student mentioned how important it was to allocate sufficient time, to avoid time pressure and stress towards the end of the lecture.

P7 mentioned specifically how she struggles when she is not able to grasp the complete picture of what is being developed during the live coding session. She emphasised the importance of conveying how the smaller bits fit into a larger whole, and that the lecturers should focus more on overall programming strategy and goals during the live coding.

The students were a bit torn in regards of the environment for the lecture. Some liked the use of an auditorium, not a class-room, otherwise you could not see properly from the back rows (P2, P10). Another student explained learning was better in settings with fewer students, compared to sitting in a large auditorium (P7). P2 pointed towards time of day as important, to be able to concentrate and to be awake to keep up. P6 stressed the importance of lighting, meaning that the lecture room needs to be dark enabling the focus to be on the screen (and code).

5 Discussion

Interesting aspects to discuss and different student strategies during the lecture emerged both from surveying related work and were made concrete through the data collection. From the interviews we learned that students seem to try out a variety of techniques for engagement, collaboration and learning during the lecture. This is in line with the findings from Bonwell et al. [9] and Weltmann [10]. All the candidates interviewed reported they copied (or had tried to copy) the lecturers code simultaneously as it was written. One filled in that this was a fine technique in the introductory course, but as the topics became more advanced this stole the focus. Another technique that were used was to take notes of interesting elements heard or seen during the lecture for then after class returning to these an investigating them with sufficient time and concentration. A majority also mentioned that their primary recipe for success was to just pay attention to what was going on. By doing this it allowed for a bird's-eye view and by this giving an overview of the code sample, and it allowed them to focus their attention to what was told by the lecturer during the walkthrough and coding.

Perceived benefits from live programming where eagerly described by the interviewed students. A common reported characteristic in our findings is that it is a beneficial way to learn. They mention it is easy to ask questions, and these can lead to new examples or demoed changes on the go, as well as interesting diversions. Together with exemplification, thorough explanations are very well received and reported to be of great use. Further to this, observing also give the students the possibility to see the bigger picture and understand strategies applied to solve a problem. As promoted by Gaspar and Langevin [15], it is more focus on the process, rather than the end product, any with this they learn more about the programming process and utilised strategies. It is also mentioned that through this they pick up a variety of tips and tricks connected to the development tools, shortcuts and other related unrelated information.

Another interesting observation from the interviews is that the audience (students) perceive this form of lecture to be interactive and active. This do not only go for the periods while they code simultaneously, but also when purely observing. The students feel they are being activated. As supported by Freeman et al. [12], active learning strategies support increased learning outcomes. Their results found this to be supported unrelated of class size and course type. Our interviews are inconclusive towards this and suggests it must be further investigated. As presented, some students preferred small classes whereas others found 1:1 learning ideal.

The interviewed students highlight speed as one important challenge. If the lecturer goes to fast it is very difficult to be able to understand the topic, comprehend flow and understand the rationale behind decisions taken during the session. What determines "too fast" is a subjective opinion, but they commonly point out that they at some stage typically drop out. This can be due to a surprising shortcut keyboard command doing a magic trick or being lost between alternating files or unable to understand an explanation. Ones lost, the interviewed students point out that it is close to impossible to regain attention and catch up.

Unsatisfactory explanations are also a challenge reported by several. It is energy-consuming to be able to follow and to be able to understand and learn. The lecturer must be able to give a precise, accurate and well-reasoned comment or explanation to the written code. If this fails, then the entered code remains without a contextual connection, and seemingly lost and without any practical opportunity to be remembered. As researched by Gaspar and Langevin [15], the learning process of programming is about

more than just syntax and detailed program code. Sometimes it is easy for students to get to fixated on the details in the code and difficult to grasp the overall picture of what are doing. Still, it is necessary to see the larger picture to understand what one is doing in the smaller scale.

We also asked if they could share some thoughts on how this lecture technique could be further improved. Several of the interviewees immediately answered that better explanations would be the first thing that comes to mind. The reason for this being that since this is well known to the lectures s/he is not always stating the obvious. These, seemingly, unimportant details are essential to the understanding, particularly for the introductory classes. The second item on the list of improvements relates back to the results reported for challenges, namely speed. The lack of speed leads to the students being bored and feeling it becomes to elementary, whereas to high speed loose the attention of the lecture room. They point out they have no solution to how to balance speed, but nevertheless it is felt like an active barrier when not matched to their liking.

6 Conclusion and Future Work

We asked the research question *from the student's perspective, what are the experienced positive and negative effects on their learning process from live programming lectures?* and researched this through gathering data using semi-structured interviews with bachelor students.

Differentiating from previous works, we sat out to investigate this question purely from the students' perspective. Additionally, we added the dimension of studying an advanced programming course in addition to an introductory course. Some of our findings such as students' engagement and desire to learn through active learning techniques are supported by previous findings [12], and the importance of understanding both the syntax and the larger picture at an architectural scale [15]. Other findings such as the huge importance of speed, the re-explanations of already covered material and attention to details were brought forth as new aspects from the students. This indicates that continues shift between lecture, demo and exercise, repeated in small iterations throughout the whole lecture, might be an interesting angle to purse and measure the effect of.

We sincerely think it is a need for more in depth results connected to the phenomena of live programming from the student perspective, and we especially think advanced courses must be researched to a greater extent. Measured achieved learning outcome and student performance are examples of two topics worthy of further pursuit.

Limitations

We acknowledge our research has some limitations. Firstly, we have a limited number of participants and they are all from the same institution. Although they have experienced the topic of live programming from multiple lecturers in multiple courses this might still bias their attitude. Secondly, being a small sample of 10 interviewed students, we cannot generalise from this study, but rather investigate, question and highlight topics for future research.

References

[1] Kidsa Koder. Lær kidsa koding. `https://kidsakoder.no`, 2018. Accessed: 02.04.2018.

[2] Computing at School Initiative. Computing at school. http://www.computingatschool.org.uk, 2018. Accessed: 02.04.2018.

[3] The Code Club Initiative. Code club. https://www.codeclub.org.uk/, 2018. Accessed: 02.04.2018.

[4] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. Success in introductory programming: what works? *Communications of the ACM*, 56(8):34–36, 2013.

[5] John Paxton. Live programming as a lecture technique. *Journal of Computing Sciences in Colleges*, 18(2):51–56, 2002.

[6] Per Lauvås and Rolando Gonzalez. An experience report using scrimba: An interactive and cooperative web development tool in a blended learning setting. *Proceedings of Norsk Informatikkonferanse*, 2017.

[7] Amy Shannon and Valerie Summet. Live coding in introductory computer science courses. *Journal of Computing Sciences in Colleges*, 31(2):158–164, 2015.

[8] Marc J Rubin. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 651–656. ACM, 2013.

[9] Charles C Bonwell and James A Eison. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.* ERIC, 1991.

[10] David Weltman. *A comparison of traditional and active learning methods: An empirical investigation utilizing a linear mixed model*. The University of Texas at Arlington, 2007.

[11] Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

[12] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415, 2014.

[13] Benjamin S Bloom. *Taxonomy of Educational Objectives: The Classification of Educational Goals: By a Committee of College and University Examiners*. David McKay, 1971.

[14] Patrick Seeling and Jesse Eickholt. Levels of active learning in programming skill acquisition: From lecture to active learning rooms. In *Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2017.

[15] Alessio Gaspar and Sarah Langevin. Active learning in introductory programming courses through student-led "live coding" and test-driven pair programming. In *International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL*, 2007.

[16] Lioness Ayres. Semi-structured interview. *The SAGE encyclopedia of qualitative research methods*, pages 811–813, 2008.