2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) | 978-1-6654-9592-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSE-SEET55299.2022.9794193

Exploring Diversity in Introductory Programming Classes: An Experience Report

Iris Groher
Michael Vierhauser
iris.groher@jku.at
michael.vierhauser@jku.at
Johannes Kepler University Linz
Linz, Austria

Barbara Sabitzer
Lisa Kuka
barbara.sabitzer@jku.at
lisa.kuka@jku.at
Linz School of Education
Linz, Austria

Alexander Hofer
David Muster
alexander.hofer@jku.at
david.muster@jku.at
Johannes Kepler University Linz
Linz, Austria

ABSTRACT

Digitization is becoming part of almost everyone's life, ranging from smartphones and tablets, smart devices automatically collecting information, to tools and scripting languages that are widely available and easy to use. This has recently been reflected in various university curricula, where courses such as computational thinking, and basic programming classes are now included in a broader range of (non-computer-science) programs. However, these programs often face challenges in such courses due to their diverse student body, with students often lacking profound digital competencies and technical background. While gender diversity aspects have been broadly studied in the past, other diversity dimensions such as ethnicity, age, or educational background have largely been neglected thus far. In this paper, we report on our experiences in teaching an introductory programming course to first-year Business Informatics bachelor students. After undergoing fundamental changes in our teaching concepts and the provided learning material, we explore what diversity factors play an important role when teaching programming to non-computer science students, and how diversity is perceived by lecturers and tutors. Our analysis confirms that a collaborative teaching concept positively supports female students and students with language barriers.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; • **Applied computing** → **E-learning**.

KEYWORDS

Introductory Programming, Diversity Dimensions

ACM Reference Format:
Iris Groher, Michael Vierhauser, Barbara Sabitzer, Lisa Kuka, Alexander Hofer, and David Muster. 2022. Exploring Diversity in Introductory Programming Classes: An Experience Report. In *44nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3510456.3514155

ICSE-SEET '22, May 21–29, 2022, Pittsburgh, P A, USA

ACMISBN 978-1-6654-9592-9/22/05...$15.00
https://doi.org/10.1145/3510456.3514155

1 INTRODUCTION

Digitization is becoming part of almost everyone's life, ranging from smartphones and tablets being used to perform tasks that were previously done with pen and paper, smart devices automatically collecting information, to tools and scripting languages that are widely available, and easy to use. One of such are, for example, Jupyter Notebooks [1], an easy to use, web-based and open-source tool, that provides a programming environment for data science applications, but also for developing simple programs and sharing educational materials. With this shift from writing programs and scripts being a skill only relevant for a small number of people in the domain of computer science (CS), to a vital asset for less technical domains, more and more non-computer-science students require technical skills to use, interact with, and sometimes even write simple programs.

This has been recently reflected in various university curricula, where courses such as computational thinking, introductory algorithm, and basic programming classes are now included in a broader range of programs, especially for non-technical fields of study, such as business administration, or business management [6, 10, 22, 30]. This in turn leads to challenges in these rather technical courses, due to their often diverse student body. Students in these courses often lack profound digital competencies and technical background, leading to high drop-out rates and moderate results by those who pass [15, 21, 32]. Business Informatics is at the intersection of these technical and non-technical fields, attracting both students with a keen interest in computer science, as well as business administration and management. Unlike the CS curriculum, which typically puts a strong emphasis on technical skills, algorithms, or formal methods, Business Informatics incorporates both technical and business aspects. While students are provided with a solid set of skills in programming, software engineering, and software architecture, an additional focus is on project management skills, information system management, and digital transformation processes in organizations. This in turn leads to a heterogeneous and diverse group of students with different interests, gender, cultural background, and educational background.

Over the past decade, we have been teaching programming principles, an introduction to programming, to first-year Business Informatics bachelor students and have observed high drop-out rates and challenges that students are facing due to their cultural and educational background [24]. While gender is still an important factor that needs to be taken into consideration in STEM education, there are, however, other aspects that influence how well students perform in this area [12, 15]. Particularly in the Business

Informatics program, we have observed that students have a very diverse educational background, ranging from extensive programming knowledge taught in high schools with a specialization in computer science, to no programming knowledge at all. This situation is exacerbated by second-chance learners, as well as a high ratio of part-time employed students. Additionally, cultural differences and language barriers pose additional challenges when teaching basic programming concepts to students.

In order to tackle these issues, reduce drop-out rates, and foster an inclusive teaching and learning style, we have introduced several new teaching concepts to actively engage with students, and create a collaborative learning experience. Alongside these changes, we also collected feedback and data from students to assess the impact of these changes and how the courses, effort, and learning experience were perceived by students. After their inception, the concepts underwent several iterations, based on feedback from students and tutors, and analyzing exercise grading and exam results.

In this experience paper, we report on four years of teaching an introductory programming class for first-semester students. We have extensively analyzed data from weekly assignments and exams, feedback from students provided via pre-and post-semester questionnaires, and conducted interviews with teaching assistants, responsible for grading exercises and lecturers teaching the class in its most recent iteration in 2020. We have observed that a collaborative teaching concept positively supports female students, students with language barriers, and students with little prior knowledge. Working besides studying does not influence drop-out. Second chance learners are not disadvantaged in our course. Based on our findings we present lessons learned and concrete recommendations for teaching introductory programming classes to heterogeneous groups of students.

The remainder of this paper is structured as follows. In Section 2, we provide a brief overview of our course setup, teaching strategies, and the evolution of the course since its introduction. In Section 3 we describe our research method, data collection, and our research questions. We then report on results of our data analysis, evaluation of student questionnaires, and the results and findings of our interviews with tutors and lecturers in Sections 4 and 5. We then discuss implications, present lessons learned and recommendations in Section 6. We finally discuss related work and conclude the paper.

2 COURSE SETUP

In the following, we provide a brief overview of our introductory programming course, its contents, and structure. The course is a mandatory course in the first semester of the Business Informatics bachelor program at our university, with around 150 to 200 students attending the course each semester. It provides basic programming principles, and an introduction to software development with Java. The Java programming language was chosen deliberately, as subsequent courses such as Advanced Software Development, Software Engineering, and Software Architecture also build on this course and require basic knowledge in Java. With regards to its contents, the course starts with a basic introduction to programming principles, covers the foundations of Java programming, and concludes with object-orientation and basic inheritance principles.

2.1 Course Structure

With 6 ECTS credits, which corresponds to approximately 150 hours of work per student per semester, the course is divided into two parts: a weekly lecture, where attendance is optional (but highly recommended), and a weekly exercise with mandatory attendance.

The lecture part is a 90-minute slide-based lecture covering one topic each week, over 14 weeks. Additionally, live-coding sessions during each lecture are used to demonstrate how programming tasks are performed, to provide students with hands-on experience. Students have access to the slides before the lecture, an optional textbook is available, and additional examples can also be downloaded. Students are encouraged to read the book chapter before the lecture and study the slides and prepare questions.

Before we changed our teaching concept to the one described below, we followed a traditional teaching approach, where both the lecture and the exercise were purely slide-based. The exercise typically takes place on the following day of the lecture and is synchronized in terms of the topics and the homework assignments that are distributed. For the exercise, students are split into groups of approximately 30 people, held in smaller seminar rooms. We regularly had quite high drop-out rates, around or above 50%, and moderate results by the students who passed our course. Also, we could observe a significant gender gap concerning both drop-out and exam results. To improve the learning outcome, and ultimately reduce drop-out rates, we decided to apply a new teaching concept in the exercise. The new concept is adapted from the teaching concept proposed by Sabitzer *et al.* [25] for STEM classes at the university level. It has a strong focus on discovery and cooperative learning [24, 26], and instead of solely relying on front-of-class teaching, the exercise is split into three parts:

(1) Repetition and Questions: At the beginning of each exercise, for about 15-20 minutes, the lecturer summarizes the most important concepts of the previous lecture and provides additional examples and code snippets. During this time, students are encouraged to ask clarification questions. It is important to note that this summary is not meant to be a replacement of the actual lecture, as only selected parts and a summary are presented, which is also clearly communicated to the students.

(2) Discovering: In a second step, another 10-15 minutes are dedicated to self-learning. Students have time to take a look at what we call "Reading Corners", where we provide sample solutions, step-by-step exercises, and examples related to the topic (e.g., different types of loops, and examples of how they are used). We provide snippets of executable code with additional comments in the code in a collaborative online Java editor [23].

(3) Pair programming: Finally, the last part, and also the majority of the 90-minute exercise (about an hour) is dedicated to teamwork and pair programming. Students work together, in groups of 2 or 3, on their weekly assignments, following the rules of pair programming as an effective method in programming education [9]. They can ask their partners for help if they experience any problems, and the lecturers act as coaches being available to answer questions and help when needed.

In the semester of 2020, due to the COVID pandemic, lectures and exercises had to be held online via Zoom. For the exercise, we tried to follow the same teaching concepts previously introduced

as close as possible. Pair programming sessions were performed in breakout rooms, and the lecturer was called for help when needed. Students shared their screens or used the collaborative online editor to work together on the assignments. Additionally, a lecturer visited each team at least once during the pair programming session.

Structure of homework assignments: In the process of introducing new teaching concepts, we also made significant changes to the weekly homework assignments. Before, each assignment consisted of two rather large and complex programming tasks (each worth half of the assignment points), often including mathematical concepts. We changed the assignments to multiple smaller examples, typically five to six individual tasks from different domains. Also, we included different types of tasks, apart from programming-only tasks, such as reading and describing code snippets. Each student has to submit the weekly homework assignment tasks electronically within one week. Based on our observations, students typically manage to finish about one third to half of the assignment during class and the rest has to be completed at home.

Out of ten assignments that are handed out, eight must be submitted. If more than eight are submitted, we only include the best eight assignments for calculating the final grade. Each assignment is manually graded by a tutor (typically a student in a higher semester, that has already completed the course and follow-up courses) that provides detailed feedback regarding the assignments, errors made by the students, and efficiency of the solutions.

In addition to the lecture and exercises, we offer a voluntary weekly tutorial. The tutorial is typically run by a tutor and he or she provides support should there any issues or problems occur before the submission. The tutorial is typically held a few days before the homework assignment is due, to give students enough time to work on the assignment at home. Same as the lectures and exercises, during the semester of 2020, the tutorial was held online via Zoom, and students were provided individual help in breakout rooms by the tutors.

Teaching materials: Besides lecture slides and homework assignments, the students receive different learning material for studying the course contents. They are provided with supporting literature in the form of books, summary slides of the exercises with additional examples, and a weekly Reading Corner that contains sample solutions and step-by-step examples to foster pattern recognition and discovery learning [25]. In addition, links to videos are provided that contain further examples and coding sessions.

2.2 Exams and Grading

As students have to take both, the lecture and exercise at the same time as part of the introduction to programming module, they receive the same grade for both classes. This means the final grade is calculated based on a combination of assignments and exam results. This should also prevent students from handing in assignments and passing the exercise part, but skipping the exam and delaying finishing the lecture part. In order to avoid that students only start learning for the final exam a few weeks before the end of the semester, as part of the new teaching concept, we introduced a mid-term exam. Students now have two options on how to pass the course: They can either participate in the shorter mid-term and end-term exam, each 45 minutes, or take the final end-of-semester exam which lasts 90 minutes. The advantage for the students hereby is that shorter exams only cover parts of the course, i.e., the mid-term exam covers lectures 1-6, and the end-term exam covers the contents of lectures 7-12, whereas the 90 minute exam covers all 12 lectures. As part of our new teaching concept, we did not change the number of tasks and topics covered in the exams. Similar to the homework assignments we used tasks from different domains and avoided mathematical concepts. To pass the course, students have to (1) receive at least 50% of the total points of either the two short exams or the end-of-semester exam, and (2) hand in at least eight of the ten homework assignments, with at least 50% of the points of the homework assignments. We do not rely on automated grading, and the course lecturers grade all exams manually (except for multiple-choice questions), and also try to provide feedback to students. Grades range from 1 to 5, with 1 being the best grade and 5 the worst.

3 METHOD

To gain insights on how well the new teaching concepts worked, we collected and analyzed both quantitative and qualitative data from our introductory programming course. Concerning diversity, we were particularly interested in exploring the different diversity factors when teaching basic programming principles, and the extent to which our course is affected by these factors. We, therefore, i) analyzed enrollment numbers and grades over four years, as well as detailed data from assignments and exams from 2020; ii) collected detailed information from students participating in the most recent class of 2020 using questionnaires; and iii) conducted semi-structured interviews with lecturers and tutors of that year. Based on this data, we answer the following three research questions:
RQ1: What is the effect of our new didactic concepts in terms of grades and gender gap? With this first research question, we investigated if there are any substantial changes and improvements in grades (in general and concerning gender gap) and overall results after introducing our new teaching concepts.
RQ2: What other diversity factors apart from gender play a role in introductory programming classes? For the second research question, we analyzed the results of exercises and exams concerning different factors such as students' gender, background, and previous education. The goal was to gain insights into if and why certain students perform better than others and/or drop out more frequently, and how this can be addressed with our teaching concepts.
R3: How is the influence of diversity factors perceived by lecturers and tutors? For the last research question we performed interviews with lecturers and tutors to identify diversity factors that are perceived relevant, and to what extent diversity influences the planning and execution of the course.

3.1 Data Collection

We collected qualitative and quantitative data from different sources including data about exercises and exams, as well as feedback from students using questionnaires, and lecturers and tutors via semi-structured interviews.

Enrollment Data, Grades, Exercise, and Exam Results: We collected enrollment data and grades from the introduction of the new teaching concepts in 2018 until 2020 and compared them

against the last year of the old concepts in 2017. For the class of 2020, we collected data from the weekly assignments, including points achieved per assignment, and points achieved for the mid-term and end-term exams. Additionally, we collected data on to what extent the provided online material (e.g., self-assessments, links to videos, or links to additional tutorials) were used by the students.

Student Questionnaires: To collect information from students throughout the course, we created three online questionnaires which we asked students to complete. The first questionnaire was sent to students as part of the first lecture (week 1), collecting sociodemographic data regarding age, gender, nationality, language skills, professional activity, educational background, prior experience in programming, and whether they had attended the course before. Furthermore, we collected data regarding interest in the topics of the course, self-concept, and how they plan to learn.

A second questionnaire was sent to students after the first midterm exam (week 7). We asked how students liked the course so far, how well they have understood the course contents, and how and to what extent they were using the provided material. We were further interested in any problems they experienced, and how much time they spent on preparing for exams and exercises. We again collected data regarding self-concept, interest in the topic of the course, and learning.

Finally, at the end of the semester (week 14) we asked students to answer a final questionnaire and provide overall feedback about the course. We asked them about the course mode (lecture, exercise, and exam), and the different material we provided. We also asked them how important they rate the topic of the course for their future career and about the knowledge gained during the course.

Interviews: We collected information from two lecturers as well as two tutors in the form of semi-structured interviews. The first lecturer was responsible for the lecture itself and one of the exercises, and the second lecturer was responsible for another exercise. The tutors were responsible for grading weekly assignments, providing feedback on the assignments, and running the weekly tutorial.

For both groups, we first collected demographic data (age, gender, and teaching experience/experience as a tutor). For the lecturers, we asked them about what diversity means in the context of teaching, and whether they consider this aspect when planning the lectures. Furthermore, we asked about diversity among colleagues and challenges regarding diversity in teaching. As a second part of the interview, we introduced them to the diversity wheel by Gardenswartz and Rowe [8] (cf. Fig. 5) and asked them if the wheel introduces new diversity dimensions they have previously not taken into consideration. We further were interested in how the different dimensions and aspects in the diversity wheel influence teaching, and whether important aspects are missing in the wheel.

For the tutor interviews, we were interested in the weekly tutorials, what type of questions students asked, how the tutors helped the students solve their problems, and if and how the types of questions did change during the semester. Same as for the lecturer interviews, we also introduced them to the diversity wheel and asked them about diversity factors in the tutorial.

3.2 Data Analysis and Interpretation

For enrollment data and grades we collected information from our course management system, anonymized the data, and combined the information with data from the student questionnaires regarding the students' gender, educational and cultural background (if the student did provide the optional student id in the questionnaire). Based on this data, two researchers analyzed students' performance, drop-out, etc. over the semester (cf. Section 4). For the interviews, one researcher created semi-structured interview guides, one for each target group. We then conducted a pilot interview, with a senior researcher and made minor adjustments to the questions to improve clarity. For the lecturers, we interviewed two lecturers who were not part of this publication and did not have any previous knowledge or experience of the diversity wheel. All interviews were conducted via Zoom, were recorded, and lasted approximately 20 minutes. Afterwards, two researchers individually carefully analyzed the recordings and extracted key statements with regards to student learning progress and diversity factors. Finally, the two researchers compared the results and discussed extracted findings. We discuss results and findings from the data analysis and interviews in the following sections.

4 RESULTS

In this section, we report results of our analysis of the new teaching concepts and diversity dimensions. We further discuss feedback from the questionnaires and interviews with lecturers and tutors.

4.1 Course History

As described in Section 2, we followed a traditional teaching approach in our introductory programming course until the winter semester of 2017. Starting from 2018, our new didactic concept was introduced and we made slight modifications and improvements (e.g., teaching and learning material) in the following years.

In the last year of the old course format in 2017, 200 students were enrolled in our class. All reported numbers are actual participants and we excluded students who were enrolled in the class but did not show up, or never handed in any of the assignments. (Negative grades can only be given to students who handed in at least one homework, or actively participated in class). From our experience, this number is rather consistent with approximately 150-200 participants every year, with a slightly lower number in

Table 1: Overview of participants, percentage of female participants, and course results from 2017-2020. (* 2017 was the last year before the new teaching concept was introduced).

Year	Part. [#]	Female [%]	Passed [%]	Mean Grade
2017*	200	33%	50%	3.65
2018	152	43%	61%	3.16
2019	137	37%	80%	2.41
2020	176	29%	73%	3.00

2019 (cf. Table 1). The rate of female students also remained fairly constant over the years with a male to female student ratio of approx. 70/30. For the last year of the old format, 50% of the students received a negative grade, which was also consistent with previous years. We noticed a gender gap with 55% of the male students and only 39% of the female students having passed the course. Overall, the mean grade was 3.65 for this course in 2017.

In the three years where we employed the new teaching concepts, from 2018 until 2020, we could observe significant improvements in terms of the raw course numbers. After the first year, we could observe that the rate of students that passed the class increased from 50% in the previous year to 61%, and to 80% in 2019. Finally, in 2020, 74% passed our course. More importantly, we could observe a positive trend in the rate of male and female students passing the class. In 2018, 64% of the male students and 55% of the female students passed, and in 2019 the rate was equal with 80% male and female students passing the course.

Based on this initial analysis, with regards to the first research question, we conclude that the new teaching concepts did in fact improve grades and rates of students passing the class. Furthermore, the gender gap in terms of negative grades was closed in 2019, after the second year. It is important to note that these changes were not due to significant changes in the contents of the course. One could argue that making the assignments or exams easier, reducing the contents, or lowering requirements can result in the desired overall improvements of the course. However, in terms of its contents and the covered topics, the slide-based lectures remained the same, we only added additional examples and live coding sessions to the lectures over the years. The topics covered in the course did not change, we even added one additional topic (dynamic data structures) to both lecture and exercise. Follow-up courses require a defined level of knowledge to be taught in our introductory programming course and thus changing the contents is not possible and the topics to be covered are defined in the course syllabus.

To gain additional insights on how the new teaching concept affects different diversity dimensions, we performed a detailed analysis of the results of the class of 2020.

In particular, we focused on two different main diversity dimensions: gender and educational background (as they were the ones we assumed in the previous semesters had the most influence on drop-out and results), and also discuss findings related to other dimensions. The analysis is based on feedback from the surveys, as well as an in-depth analysis of homework assignment results and exam results of the semester.

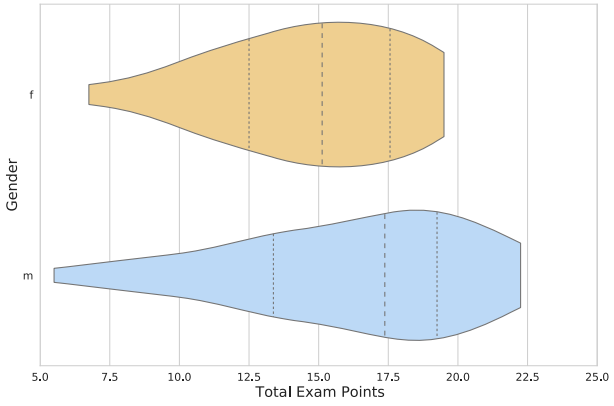


Figure 1: Distribution of average exam points by genders

The first questionnaire was handed out during the first lecture and was returned by 168 students, resulting in a response rate of 96%. The second questionnaire was sent to students after the first mid-term exam. It was filled out by 114 students, which corresponds to a response rate of 65%. Finally, the third questionnaire was handed out during the last lecture of the course and was returned by 52 students, which corresponds to a response rate of 30%.

In 2020, 73% of the students passed our course (cf. Table 1) with a mean grade of 3.0 for the course. In total, 73% of the male students and 74% of the female students passed the course.

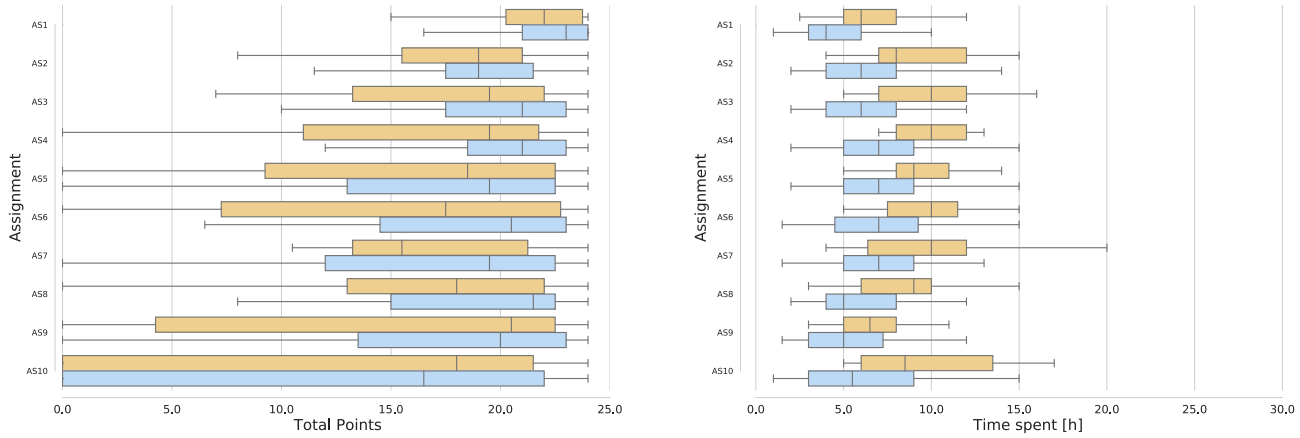
Compared to the previous year, the percentage of the students who passed the course is slightly lower and the mean grade is higher. Also, the rate of male and female students who passed is slightly different. The course in 2020 was held fully online via Zoom sessions for both the lecture and exercises, due to COVID-19, which is a possible explanation for the reduced success rate.

4.2 Gender Dimension

As we have observed a noticeable gender gap before changing the teaching concepts and a reduction thereof afterwards, we were interested in the factors that contributed to this change and if there are further improvements that can be made.

In particular, we looked at the exam results and assignment results with respect to gender differences. Fig. 1 shows the exam results of female (top orange part) and male (bottom blue part) students. The results are based on the average points students received for the shorter mid-term and end-term exams, and where gender information was available. Each exam has a maximum of 24 points and passing the exams requires an average of at least 12 points. Based on the analysis of the exam results we could observe that male students received slightly more points on average than female students (16.23 vs. 14.70). However, more female students are in the mid point range, ranging from 12.5 and 17.5 points, whereas for the male students we could observe students with fewer than 7 points and also more than 20 points.

Passing the course requires a positive grade on the exam part as well as on the homework assignment part. Regarding the homework assignments, on average female students appear to invest more time than male students, but still fall slightly behind in points achieved for the assignments. Based on the self-reported time and effort spent on each homework, female students spend on average 8.93 hours per assignment, and male students 6.62 hours. Male students achieved on average 18.86 points per assignment, while female students achieved 17.88 points. Fig. 2 shows a comparison of the distribution of points and the self-reported time spent over the ten assignments. For assignments 6, 7, and 8, the average points achieved by female students is considerably lower than the points of the male students. Assignments 6 and 7 focus on arrays, while assignment 8 focuses on object orientation. The average of assignments 9 and 10 (also focusing on object orientation) are a bit higher for female students.



(a) Comparison of distribution of points for assignments

(b) Comparison of self-reported time spent per assignment

Figure 2: Gender Dimension Assignments – male (blue)/female (orange)

This suggests that female students have difficulties with the topic of arrays and need more time for the topic of object orientation.

In summary, our new teaching concepts have almost closed the gender gap in our course. We thus conclude that a collaborative teaching concept positively supports female students. Still, more, and also different learning material for the topics they have difficulties with could be beneficial especially for female students. For example, it could help to provide videos with step-by-step examples for the topics of arrays and object orientation.

4.3 Educational Background Dimension

Another diversity dimension we observed in previous years, is the educational background of students. As Business Informatics typically attracts students with more diverse educational background, our assumption was, that this also had an effect on the prior knowledge of students when starting their first semester at the university.

From the survey results, we could confirm that the educational background of our students is in fact quite diverse. Fig. 3 provides an overview of the six different types of educational background. This includes five different types of high schools (typically with a particular focus on a certain area), as well as second-chance learners (SCL). This category groups all participants that did not obtain a high school diploma, which is the requirement for enrolling in a university course but have passed a university admissions equivalency test, which accounts for almost 20% of the students. The majority of students, almost 35% graduated from a general secondary school, which is either 8 years (GS-HS, 26.5%), or 4+4 years (UL-HS, 8.2%), sharing a similar focus in terms of broad general education with no specialization. The single largest group of students graduated from a commercial academy (C-HS), which is a school with a special focus on economy and business (28.6%). Similar to this school is the economic academy (EC-HS) (6.1%). Only 11.2% of our students graduated from a technical academy (T-HS). However, it is important to note that students who graduated from a school with a focus on CS are exempted from our course.

Despite the diverse types of high schools our students attended, 78% reported that they had some form of CS lessons during their four years of high school. However, asking students about their specific experiences with CS classes in high school revealed that the vast majority of students only focused on office tools (text processing, spreadsheets), with only a third that had obtained basic programming knowledge during these classes, with two-thirds having no prior programming experience whatsoever.

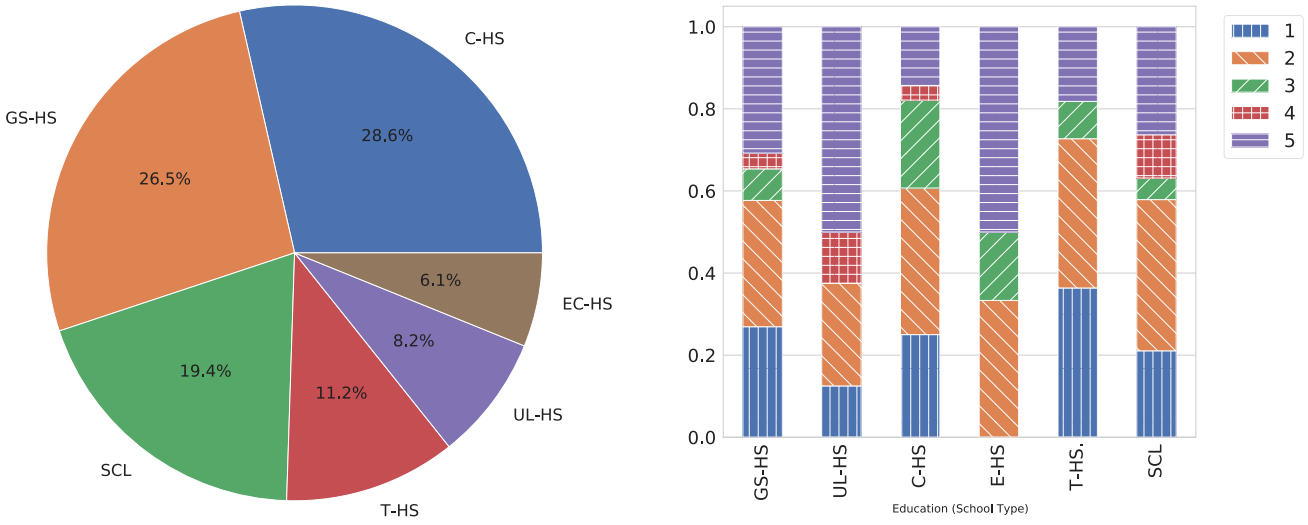
As part of this diversity dimension, we were particularly interested if students with a certain background were more likely to fail our course than others, and if there is a statistically significant correlation between educational background and passing or failing our introductory programming class.

Fig. 3 shows the grades of the students in our course for the different educational backgrounds. The data reveals that, unsurprisingly, students who graduated from a technical school performed best. Graduates from an upper-level secondary school and an economic academy performed the worst. What is interesting is that students from a general secondary school performed similarly to second-chance learners. A possible reason for this could be that second-chance learners might have worked in a technical profession, or due to their professional experience, their time management is better. However, we could not detect a statistical significance for a correlation between educational background and grades.

4.4 Other Dimensions and Observations

Besides the two major aspects of gender and educational background, the surveys contained a number of additional questions with regards to the students' background and their learning experience throughout the semester.

4.4.1 Personality Dimension & Work Experience. Results of the first survey indicated that 60% of the students work, at least part-time, and more than 20% of them work more than 20 hours per week. To support part-time and working students, we typically offer one exercise in the early morning, and another one in the late afternoon,



(a) Different School Types

(b) Course Results grouped by School Types

Figure 3: Educational Dimension Schools Types and Course Results

so that students can attend class before or after their job. The fact that in 2020 both lecture and exercise classes were held online further supported working students, as it was not necessary to attend the class in person at the university.

To gain insights into the personality dimension, we asked students about their general interest in the topic. At the beginning of the course, most students claimed to be interested in the topic of programming, with 75.9% claiming to be very interested or interested. This shows us that, even though Business Informatics has a different focus than CS, programming is not seen as something that is only a side topic, but is a major factor why students choose to enroll in the Business Informatics bachelor program.

4.4.2 General Course Difficulty, Concepts, and Learning Material. At the beginning of the semester, the course was considered difficult, only 9.6% rated the course as easy or very easy. This number tripled until the first mid-term exam, where 27.1% rate the course as easy or very easy. There is also an increase in interest from the beginning of the course to the mid-term exam. The rating of very interested increased from 35.8% to 60%.

The self-reported time invested per week differs greatly among students. 30.1% claimed to have invested less than one hour, 36.1% invested between one and three hours, 18.1% between four and six hours and 15.7% invested more than six hours per week.

Regarding the provided learning material, most students would like to have more material for object orientation and dynamic data structures. They would also like to have more exercise and lecture units for these topics. Students were happy with the material provided and time spent on the other topics.

At the end of the course, students gave positive feedback. In general, students liked the course. 75.5% rated the course (lecture and exercise) as good or very good and 81.6% rated the exam mode as good or very good. 57.1% rated the time spent as high or very high and 28.5% rated the course as difficult or very difficult. Most students (90%) claimed that they have strongly expanded their knowledge and 76% rate the importance of programming for their professional career as important or very important.

One aspect we were concerned about when introducing pair programming was that this would lead to situations where only one student would have a positive learning experience by performing the majority of the work in the team. The other team members only take a passive role in these cases.

In previous years, where each homework assignment had to be handed in individually, we performed similarity checks and graded assignments with 0 points when similarities and copy-paste of code parts were evident. With group programming activities this was

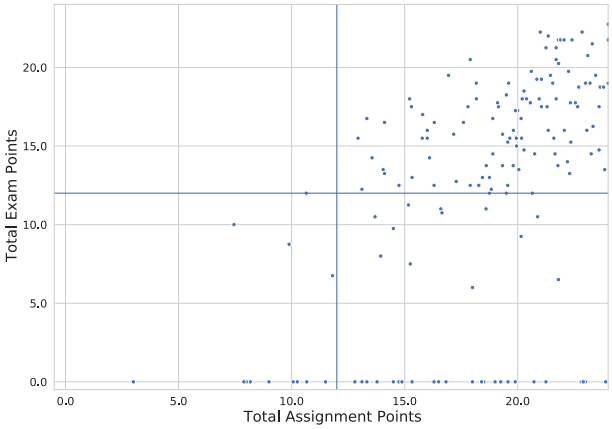


Figure 4: Comparison of assignment vs. exam points

significantly more difficult and it was easier for students to hand in exercises without fully solving them by themselves.

To prevent students from not actively engaging in the programming activities, we introduced the split mid-term and end-term exams to force students to prepare for a test after the first half of the class. The analysis of the exam points in correlation with points received for the assignments indicates, however, that the pair programming activities were an effective strategy, and did not result in degraded exam grades. Fig. 4 shows a comparison of exam results (average of mid and end-term exam), with a maximum of 24 points, and assignment points (average of best eight), with also a maximum of 24 points. The data points at the bottom represent students who handed in at least some of the assignments but never took part in any of the exams.

Based on the results from the gender and educational background analysis and the analysis of other dimensions, with regards to RQ2, we conclude that cooperation and peer learning, as well as different kinds of material, positively support heterogeneous groups and especially those with a lack of prior knowledge. Working besides studying does, in our case, not influence drop-out, and second-chance learners are not disadvantaged in our course. In general, students maintain interest throughout the semester and gave positive feedback regarding the course format.

5 INTERVIEWS

In addition to the course data we collected from the students, we also wanted to hear the other side of the story, how lecturers and tutors think about and perceive diversity factors. We, therefore, interviewed two lecturers and two tutors from the course of 2020. The lecturers were both males. The first lecturer was 33 years old with two years of teaching experience and the second lecturer was 60 years old with more than 30 years of teaching experience at different universities. The first tutor we interviewed was female and the second tutor was male. Both were 20 years old and had one year of experience as a tutor in an introductory programming course.

In the interviews we focused on i) our teaching approach in general and ii) on the different diversity aspects in particular.

In general, diversity factors were considered important by all four interviewees but did not directly influence how they were preparing and planning the lectures or the weekly tutorial. However, diversity was taken into account during the course, i.e., during lectures, exercises, or the tutorial. For example, one tutor said that she translated parts of the task description into English during the tutorial for a student with language barriers.

Diversity Factors: When asking about diversity in teaching, diversity of students, lecturers, and tutors, the most prominent factor was *Gender*, mentioned by 3 out of 4 interviewees. This was also the prevalent topic (with an emphasis on female students in STEM in general, and CS in particular) during the diversity discussion before we introduced them to the other diversity dimensions part of the diversity wheel. This comes as no surprise, as gender equality has received significant attention in the past [5, 19, 20]. Among the aspects mentioned, gender-neutral language and avoiding male and female stereotypes in lecture and exercise materials were deemed as very important by both lecturers. Additionally, having both male and female lecturers, as well as tutors, was mentioned by one lecturer, saying that *"[...] it is important to also have female teachers and tutors, as role models for girls/women [...] when starting their studies"*.

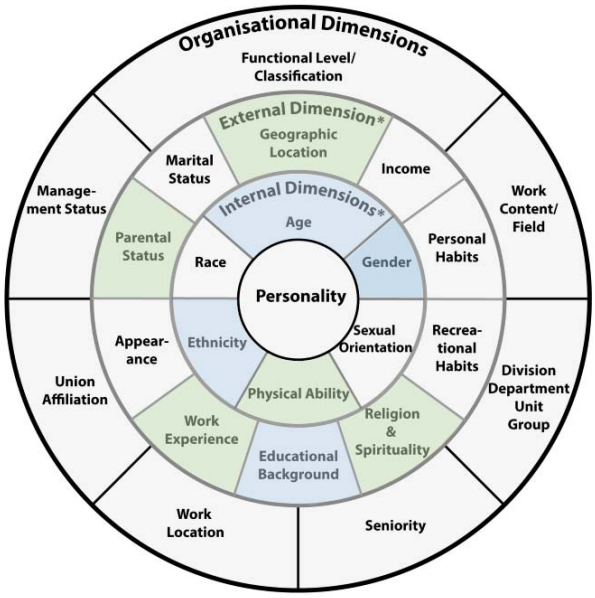


Figure 5: Diversity Wheel [8] – Dimensions colored in blue were mentioned before showing the wheel, dimensions colored in green afterwards.

Furthermore, the *Ethnicity* dimension, specifically, in the context of language, and resulting language barriers was mentioned three times, twice by the tutors and one time by a lecturer.

Additionally, one tutor also mentioned *Age* and *Educational Background* of students, which was largely overlooked by the other three. The tutor mentioned challenges faced by students with no prior programming knowledge. She also mentioned that for her it was strange to "teach" a student that was much older than her.

After showing them the Diversity Wheel, and briefly explaining the dimensions, all acknowledged that there are additional factors and dimensions with regards to diversity that they did not directly consider in the context of "Diversity", or that they have previously not thought about. Among these, the interviewees mentioned *Geographic Location* and *Parental Status* (supported through online lectures), *Work Experience* (prior knowledge and better time management), and *Religion* (sensible language in assignments). Interestingly, none of the interviewees mentioned *Personality* as a factor influencing motivation, interest, or learning strategies.

Fig. 5 shows the Diversity Wheel we presented during the interviews. Factors mentioned before the wheel was shown are marked in blue. Factors mentioned after the wheel was shown in green.

Teaching Factors: Besides the diversity dimensions, we were generally interested in how the new teaching concept was perceived by the tutors and what additional insights we could gain from their experiences during the weekly tutorials. One interesting aspect the

tutors reported was that students had difficulties with the formal requirements of the homework assignments. Students had problems submitting the assignments as specified following the submission guidelines (one zip file per assignment with strict rules on what it must contain). Also, students had difficulties following naming conventions and code formatting rules. We insisted on these rules as follow-up courses in the curriculum have even stricter rules due to unit tests and automatic grading.

Another issue that was mentioned was the ability to fully understand the programming tasks part of the assignments. This was perceived as sometimes being quite challenging for the students. One tutor reported that students complained that the assignment descriptions were not specific enough, sometimes leaving room for interpretation. For example, it was not always clear what kind of error handling was expected, and what kinds of invalid input needs to be dealt with. This was quite surprising for us as, compared to the old teaching format, the new assignment descriptions contained much more details regarding what data types to use, what method parameters to add, or what return values to provide.

However, based on this, for students with little programming experience, step-by-step instructions are easier to follow and help to guide them through the assignment and programming tasks. There is a trade-off between supporting beginners through guidance and teaching them to develop solutions and algorithms on their own. A possible solution might be to include one task in each assignment that requires more creativity and provides less guidance.

One final aspect that was mentioned by the tutors was again due to the COVID-related shift to online classes. The lack of physical interaction was perceived as challenging. While support was provided and questions were answered during the tutorials via Zoom, the tutors mentioned that it was sometimes hard for them to see if students had in fact fully understood their explanations on how to solve a problem. Specifically, the lack of gestures, or how students behave or ask questions in a physical, in-person tutorial not only helps the lecturer during class to see if someone needs help, but also negatively affects tutorials.

With regards to our third research question, we can conclude that gender was by far the most important factor that is associated with the term diversity. Nationality and related language barriers are also regarded as important. However, when being presented with the whole spectrum of diversity dimensions, all interviewees identified additional dimensions and stated that when they would have known the wheel before, they would have thought differently about diversity. We thus conclude that educating teaching personnel with respect to diversity and its multiple dimensions will bring a broader view on this topic into university lectures.

6 DISCUSSION AND LESSONS LEARNED

Based on the results and findings from our analysis, and the additional data we gained from the interviews we discuss the impact on diversity and provide a number of recommendations.

6.1 Diversity is more than Gender

In our introductory programming course, we are faced with a quite heterogeneous group of students, pertaining to several different factors. Analyzing the demographic data of our students revealed that, apart from gender diversity, students have diverse educational background, work experience, prior knowledge, and nationality. Both lecturers and tutors have also confirmed that our students are quite diverse and this has an impact on how lectures are held and how tutorials are conducted. Some dimensions were immediately obvious and part of people's mindset when thinking about diversity, others were only recognized after engaging in a broader discussion. In total, nine of the dimensions were considered to play a relevant part in teaching by our interview participants.

Experienced students can act as peer tutors in the course and help beginners during pair programming. A high percentage of working students can be supported with additional online material or videos. Language barriers can be bridged by translating assignment tasks (partially) into English or by providing feedback in English. Using gender-sensitive language should, of course, be used, but solely changing the language does not solve all problems.

Recommendation 1

As one can only support what is known, we recommend collecting basic demographic data at the beginning of the course. It is important to know about the background of the students and their prior knowledge.

6.2 Diversity-aware Planning and Preparations

All of our interview participants, both tutors and lecturers, mentioned that they did not consider diversity aspects in advance during planning and that they are more important during class when interacting with students. However, certain things, such as proper teaching material, do require planning and preparation before the lecture, or even before the semester. To support heterogeneous groups of students and respond to diverse needs and abilities, a portfolio of different teaching and learning material is needed. In our course, we provided slides, books, videos, and Reading Corners with step-by-step solutions. The examples should require little mathematical background. Also, we recommend using examples where students can be creative and bring in their own ideas. We further recommend using more but smaller examples instead of a few large ones. This gives students a sense of positive achievement when they manage to complete (at least some of) the tasks.

Recommendation 2

We recommend diverse teaching and learning material. This includes different types of material, such as videos, examples, and slides, but also examples from a variety of domains.

6.3 Interactive Teaching and Learning supports Diversity

In our exercises, a majority of the time (approximately two-thirds) is dedicated to group work and pair programming sessions. This fosters active student participation and allows the lecturers to get first-hand experience of how students work on assignments, what difficulties they experience, and what common mistakes are made.

This in turn helps to improve, and iteratively refine teaching material and instructions, to explain and prevent common mistakes in the future. The feedback we have received with respect to pair programming from the students was very good. The data we collected shows that there is a correlation between exercise points and exam points which shows that typically students do not simply copy the solution of their team partner. Or if they do so, they still learn enough to pass the exam. This further supports diverse levels of prior knowledge as students can help and support each other during the coding sessions. Supporting others also helps the experienced students as they further deepen their knowledge when teaching their peers. Language barriers can also be positively affected. The ideal situation would be to create diverse pairs with experienced and non-experienced students.

Recommendation 3

We recommend adopting active learning and communication in programming classes. Teamwork and cooperation seem to positively support female students. Furthermore, when lecturers act as coaches during the coding sessions, this helps them to better understand the difficulties of the students.

6.4 Tutors are Links between Lecturers and Students

We provide weekly tutorials for our students to ask questions about their assignments, problems they experience, or the points and feedback they received for previous assignments. All assignments that are handed in are graded and corrected by tutors. Students not only receive points for assignments but also get weekly feedback on mistakes they made or guidelines they did not follow. Students regarded the feedback and the weekly tutorials as very helpful.

Recommendation 4

We recommend installing peers that act as tutors for beginners programmers. Students rather talk to peers about their problems and difficulties than to lecturers. We had regular meetings with the tutors to get feedback regarding the exercises, the provided material, and the challenges students faced.

7 THREATS TO VALIDITY AND LIMITATIONS

A threat to construct validity comes from the nature of the interviews we conducted with tutors and lecturers and the questionnaires we distributed to students. Answers from participants and students are subjective and may not properly reflect the actual situation. Also, the fact that we conducted the study during the COVID pandemic might have influenced the results.

We interviewed lecturers that had several years of teaching experience and that did not participate in this publication or other preparations for the study. The interviews have been performed by a PhD student who was not part of the lecture, nor did she work at the institute of the lecturer. The semi-structured interviews allowed us to clarify answers from participants if needed, and to ask for details on certain diversity aspects. Furthermore, two researchers independently transcribed the interviews and extracted statements. All results and diversity dimensions mentioned by interviewees were discussed extensively among the two researchers.

The experiences and perceptions of this student population may differ from the ones of other students in the same or other institutions, countries, and cultures. Additionally, students who consent to participate in interviews about group work and have their answers used in research projects may not reflect the general student population.

Furthermore, our study is limited to one university and the number of students participating in the course. Data covers four years, with three of them covering our new teaching concepts, and one year of detailed analysis. The reported experiences and perceptions of the students might differ from other students who did not participate in the surveys or from students in the same or other institutions. With regards to internal validity, a threat is the social pressure that the students might have felt when providing information regarding perceptions about the class, perceived difficulty, and teaching concepts. However, we experienced that students generally appeared comfortable discussing issues during the exercises and mention positive as well as negative aspects of the course, teaching materials, and teaching concepts. Additionally, as our results and findings yield similar results as other studies, we are confident that some observations and our lessons learned are generalizable for other introductory programming courses.

8 RELATED WORK

Several authors have discussed gender differences in learning styles [14, 16] and cultural differences [11]. Gender-sensitive learning environments and teaching concepts have been identified as important for young women in CS [27, 28].

The lack of women in CS programs at universities has been studied by several authors. For example, Cheryan *et al.* [3] and Master *et al.* [17] investigated gender stereotypes related to computing and uncovered that these act as "gatekeepers" preventing women from choosing CS studies. Similarly, Borsotti [2] has investigated barriers for gender diversity in Software Development Education. She empirically investigates the main barriers for female participation in CS programs and discusses insights and interventions. Similar to our findings, she identifies the lack of CS education at school as one important factor. In another study, Margolis and Fisher [7] managed to increase female student numbers, based on their findings from a study in an undergraduate CS program.

Other work deals with high failure rates of introductory programming courses. In this domain, Vihavainen *et al.* [31] have performed a systematic review of 60 approaches that try to improve the performance in such courses. The results showed that on average, new teaching formats can improve success rates in programming classes by nearly one-third when compared to traditional lectures. This is also confirmed by the improvements in our course after introducing our new teaching concepts.

Several works have introduced measures and strategies that are also included in our teaching and learning concepts and materials. Van der Meulen and Aivaloglou [29] introduced group assignments and pair programming and investigate work division and allocation strategies in group work. Corno *et al.* [4] propose so-called Code Recipes, consisting of summarized and well-defined

documentation modules which can be compared to our reading corners. Krusche *et al.* [13] have investigated methods for teaching modeling by providing guided tutorials for lectures, group work, and homework assignments. Particularly, pair programming is frequently applied in novice programming courses [18, 34] and studies have confirmed that it specifically supports female students [9, 33]. Koulouri *et al.* [12] show in their study that teaching problem-solving before programming yielded significant improvements in student performance. This is something we could also include in our introductory programming course as part of future improvements and course updates.

9 CONCLUSION

In this paper, we reported on introducing new teaching concepts to a first-semester programming course. This study aimed to explore the experiences and diversity dimensions that impact teaching and learning experiences. As part of the study, we analyzed data from four years and conducted a detailed analysis of the last year of our course with questionnaires and interviews. The results indicate that besides gender, the educational background, as well as the work experience of students play an important role. Based on this detailed analysis we discussed lessons learned and recommendations for teaching introductory programming classes to non-computer science students. Furthermore, our analysis has shown that a collaborative teaching concept supports female students as well as students experiencing learning difficulties due to language barriers.

ACKNOWLEDGMENT

The work in this paper has been funded by the JKU Business School and the Linz Institute of Technology (LIT-2019-7-INC-316).

REFERENCES

[1] 2021. Project Jupyter. https://jupyter.org [Last accessed 01-01-2022].
[2] Valeria Borsotti. 2018. Barriers to Gender Diversity in Software Development Education: Actionable Insights from a Danish Case Study. In *Proc. of the 2018 IEEE/ACM 40th Int'l Conf. on Software Engineering: Software Engineering Education and Training*. IEEE, 146–152.
[3] Sapna Cheryan, Allison Master, and Andrew N. Meltzoff. 2015. Cultural stereotypes as gatekeepers: increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology* 6 (2015), 49.
[4] Fulvio Corno, Luigi De Russis, and Juan Pablo Sáenz. 2018. Easing IoT Development for Novice Programmers through Code Recipes. In *Proc. of the 40th Int'l Conf. on Software Engineering: Software Engineering Education and Training*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/3183377.3183385
[5] Joyce Ehrlinger, E Ashby Plant, Marissa K Hartwig, Jordan J Vossen, Corey J Columb, and Lauren E Brewer. 2018. Do gender differences in perceived prototypical computer scientists and engineers contribute to gender gaps in computer science and engineering? *Sex roles* 78, 1 (2018), 40–51.
[6] Juan Carlos Farah, Arielle Moro, Kristoffer Bergram, Aditya Kumar Purohit, Denis Gillet, and Adrian Holzer. 2020. Bringing Computational Thinking to non-STEM Undergraduates through an Integrated Notebook Application. In *Proc. of the 15th European Conf. on Technology Enhanced Learning*.
[7] Allan Fisher and Jane Margolis. 2003. Unlocking the Clubhouse: Women in Computing. *SIGCSE Bull.* 35, 1 (Jan. 2003), .23.
[8] L. Gardenswartz and A. Rowe. 1994. *Diverse Teams at Work: Capitalizing on the Power of Diversity*. Irwin Professional Pub.
[9] Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. 2011. Pair programming in education: a literature review. *Computer Science Education* 21, 2 (2011), 135–173. https://doi.org/10.1080/08993408.2011.579808
[10] Hsing-Yu Hou, Somya Agrawal, and Chin-Feng Lee. 2020. Computational thinking training with technology for non-information undergraduates. *Thinking Skills and Creativity* 38 (2020), 100720.
[11] Simy Joy, Simy Joy, and David A. Kolb. 2009. Are there cultural differences in learning style. *International Journal of Intercultural Relations* (2009). https://doi.org/10.1016/j.ijintrel.2008.11.002
[12] Theodora Koulouri, Stanislao Lauria, and Robert D. Macredie. 2015. Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches. *ACM Trans. Comput. Educ.* 14, 4, Article 26 (Dec. 2015), 28 pages. https://doi.org/10.1145/2662412
[13] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. 2020. An interactive learning method to engage students in modeling. In *Proc. of the ACM/IEEE 42nd Int'l Conf. on Software Engineering: Software Engineering Education and Training*. 12–22.
[14] Sadan Kulturel-Konak, Mary Lou D'Allegro, and Sarah Dickinson. 2011. Review Of Gender Differences In Learning Styles: Suggestions For STEM Education. *Contemporary Issues in Education Research* (2011). https://doi.org/10.19030/cier.v4i3.4116
[15] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A Study of the Difficulties of Novice Programmers. *SIGCSE Bull.* 37, 3 (June 2005), 14–18. https://doi.org/10.1145/1151954.1067453
[16] Wilfred W. F. Lau and Allan H. K. Yuen. 2010. Gender Differences in Learning Styles: Nurturing a Gender and Style Sensitive Computer Science Classroom. *Australasian Journal of Educational Technology* (2010). https://doi.org/10.14742/ajet.1036
[17] Allison Master, Sapna Cheryan, and Andrew Meltzoff. 2016. Computing Whether She Belongs: Stereotypes Undermine Girls' Interest and Sense of Belonging in Computer Science. *Journal of Educational Psychology* 108 (04 2016).
[18] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The Effects of Pair-Programming on Performance in an Introductory Programming Course. *SIGCSE Bull.* 34, 1 (Feb. 2002), 38–42.
[19] Paola Medel and Vahab Pournaghshband. 2017. Eliminating gender bias in computer science education materials. In *Proc. of the 2017 ACM SIGCSE Technical Symp. on Computer Science Education*. 411–416.
[20] Dee Michell, Claudia Szabo, Katrina Falkner, and Anna Szorenyi. 2018. Towards a socio-ecological framework to address gender inequity in computer science. *Computers & Education* 126 (2018), 324–333.
[21] Iain Milne and Glenn Rowe. 2002. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies* 7, 1 (March 2002), 55–66. https://doi.org/10.1023/A:1015362608943
[22] Juena Ahmed Noshin and Syed Ishteaque Ahmed. 2018. Teaching programming to non-Programmers at undergraduate level. *International Journal of Engineering and Management Research (IJEMR)* 8, 3 (2018), 191–194.
[23] Replit. 2021. https://repl.it [Last accessed 01-01-2022].
[24] Barbara Sabitzer, Iris Groher, Johannes Sametinger, and Heike Demarle-Meusel. 2020. COOL Programming: Improving Introductory Programming Education through Cooperative Open Learning. In *2020, Proc. of the 9th Int'l Conf. on Educational and Information Technology*. ACM, 95–101. https://doi.org/10.1145/3383923.3383943
[25] Barbara Sabitzer, Iris Groher, Johannes Sametinger, and Heike Demarle-Meusel. 2020. COOL Programming: Improving Introductory Programming Education through Cooperative Open Learning. In *Proc. of the 9th Int'l Conf. on Educational and Information Technology*. ACM, New York, NY, USA, 95–101.
[26] Barbara Sabitzer and Sandra Strutzmann. 2013. Brain-Based Programming: A New Concept for Computer Science Education. In *Proc. of the 18th ACM Conf. on Innovation and Technology in Computer Science Education*. ACM, New York, NY, USA, 345. https://doi.org/10.1145/2462476.2488328
[27] Sigrid Schmitz and Katrin Nikoleyczik. 2009. Transdisciplinary and gender-sensitive teaching: didactical concepts and technical support. *International Journal of Innovation in Education* 1 (01 2009).
[28] Bernadette Spieler and Wolfgang Slany. 2018. Female Teenagers and Coding: Create Gender Sensitive and Creative Learning Environments. In *Constructionism 2018: Constructionism, Computational Thinking and Educational Innovation*.
[29] A. van der Meulen and E. Aivaloglou. 2021. Who Does What? Work Division and Allocation Strategies of Computer Science Student Teams. In *Proc. of the 2021 IEEE/ACM 43rd Int'l Conf. on Software Engineering: Software Engineering Education and Training*. IEEE Computer Society, 273–282. https://doi.org/10.1109/ICSE-SEET52601.2021.00037
[30] Gregory Vial and Bogdan Negoita. 2018. Teaching programming to non-programmers: The case of python and Jupyter notebooks. (2018).
[31] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In *Proc. of the Tenth Annual Conf. on Int'l Computing Education Research*. ACM, New York, NY, USA, 19–26.
[32] Christopher Watson and Frederick W.B. Li. 2014. Failure Rates in Introductory Programming Revisited. In *Proc. of the 2014 Conf. on Innovation & Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 39–44. https://doi.org/10.1145/2591708.2591749
[33] Linda L. Werner, Brian Hanks, and Charlie McDowell. 2004. Pair-Programming Helps Female Computer Science Students. *J. Educ. Resour. Comput.* 4, 1 (March 2004), 4–es. https://doi.org/10.1145/1060071.1060075
[34] Laurie Williams and Richard L. Upchurch. 2001. In Support of Student Pair-Programming. *SIGCSE Bull.* 33, 1 (Feb. 2001), 327–331.