On the Effect of Onboarding Computing Students without Programming-Confidence or -Experience

Paweł Grabarczyk
CCER: Center for Computing Education Research
IT University of Copenhagen
Copenhagen, Denmark

Sebastian Mateos Nicolajsen
CCER: Center for Computing Education Research
IT University of Copenhagen
Copenhagen, Denmark

Claus Brabrand
CCER: Center for Computing Education Research
IT University of Copenhagen
Copenhagen, Denmark

ABSTRACT

Previous work demonstrates that students without prior programming experience are worse off than their programming experienced peers in terms of both Introductory Programming (CS1) grades and dropout rates. Many universities, therefore, offer an onboarding (CS0) course aimed at bridging the programming experience gap by teaching the basics of programming to inexperienced students.

This paper reports on the effects of providing a three-day elective onboarding course over a period of five years (2016–2020), involving a total of N=798 software development students at the IT University of Copenhagen. The paper compares 271 students who attended versus a baseline of 527 who did not attend the onboarding course.

The results show that programming inexperienced students are indeed able to "catch up" to the level of their experienced peers both in terms of CS1 grades and dropout rates. Aside from objectively increasing competence, the onboarding also increases confidence, self-efficacy and diminishes insecurities, according to onboarded participants. Finally, the results suggest that onboarding has the potential to increase the diversity of students.

CCS CONCEPTS

• **Social and professional topics** → **Computing education programs**; **Gender**.

KEYWORDS

software development, computer science, computing, programming, education, onboarding, gender, diversity, dropout, grades, competence, confidence, self-efficacy

ACM Reference Format:
Paweł Grabarczyk, Sebastian Mateos Nicolajsen, and Claus Brabrand. 2022. On the Effect of Onboarding Computing Students without Programming-Confidence or -Experience. In *22st Koli Calling International Conference on Computing Education Research (Koli Calling '22), November 17–20, 2022, Koli, Finland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

Koli Calling, November 17–20, 2022, Koli, Finland

ACM ISBN 978-1-4503-8488-9/22/11…$15.00
https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

Contemporary computer science students come to universities with different backgrounds, competencies, and varying levels of confidence. The lack of unified high school computer science education and the variety of online materials students could encounter before they enter the university makes it impossible to predict their level of experience. Since computer science education tries to attract more diverse groups of students, we can only expect the discrepancy between levels of experience to increase.

The difference in initial experience level is something we should avoid on its own as it has been shown to affect later performance of students [1, 17, 19]. Studies show that prior programming (in)experience has an impact on the future academic record of the students. This is especially visible in the early stages of their academic education [8, 17, 20, 21], but can sometimes extend further [1]. Additionally, difference in initial experience level may negatively impact self-efficacy of the less experienced students as comparison with peers is the foundation for their self-efficacy perception [23]. Self-efficacy is defined as a person's own belief in their ability to succeed [23]. Low self-efficacy has been found to contribute to high dropout rates in computing and teaching should therefore actively try to increase self-efficacy of students [29]. Thus, an initiative which allows peers without experience to meet and gain initial experiences with programming will positively increase self-efficacy of the student.

Apart from being a problem in and of itself, the lack of experience and confidence has an indirect impact on another big issue related to computer science education, namely: *gender imbalance*. As can be seen in [7, 35], there is a visible correlation between the perceived self-efficacy and gender. Even though there are no important differences at the level of typical computer operation, such as browsing the web or utilizing office applications, males tend to be more confident when it comes to programming and, especially, robotics [27]. The upshot of this is that any action aimed at candidates who perceive themselves as inexperienced has an indirect impact on gender diversity.

Gender imbalance is a well-known problem in IT and there is no doubt that one of the roots of this problem is the relative lack of women opting for an IT education. It is therefore not surprising that many universities employ programs that aim to change this unsatisfactory trend. It seems safe to say that there is no single, universally recognized "silver bullet" solution that could turn the tides and that many, sometimes fragmentary actions should be undertaken jointly. One of the tools for attracting women to the IT study programs is onboarding (aka, CS0) courses that aim to bridge the experience and competence gap by introducing programming

inexperienced students to the basics of programming. In this paper, we take a closer look at an elective three-day onboarding (CS0) course called BootIT which has been offered to new students a couple of weeks before the university starts, at IT University of Copenhagen, since 2016. We show that the effects of the course and the feedback provided by the students can serve as a useful illustration of the impact this type of initiative can have on the unification of the freshmen cohort by boosting the competence and confidence of those who chose it. In addition to this, the recognition of the existence of such initiatives serves to broaden the recruitment spectrum as it reassures students who may have doubts about their abilities that they are indeed very welcome and eligible for a career in computing; they will not be ostracized for lacking experience in programming.

2 THE ONBOARDING COURSE

BootIT is a three day programming course offered at the IT University of Copenhagen (ITU) to students who are enrolling in the Bachelor's programme Software development. Within this programme, students initially face the CS1 equivalent 15 ECTs[1] course, *Introductory programming*. Herein students familiarise themselves with object-oriented programming with an emphasis on modelling reality, using the "Objects First" approach [3] and a four week project. BootIT is constructed to complement this course. The BootIT course covers fundamental programming constructs in depth which are also part of the curriculumn of the CS1 course and its first four lectures. However, the fundamental concepts are at the center of BootIT allowing inexperienced students to become familiar with their semantics and syntax in a way which the CS1 course does not facilitate. On the first day of BootIT, students are introduced to the basics of imperative programming; in particular, constants, variables, assignments, and operators. On day two, students are introduced to control structures; in particular, conditional and iteration statements. The third day focuses on further programming techniques; in particular, Computational Thinking and Object-Orientation. On the last day, the students also create a project in small groups. Towards the end of the course, students are asked to evaluate the course quantitatively and qualitatively. Each day of the course is 6-7 hours of exercises and lecturing. The entire course material is available on this link.[2]

2.1 The profile of BootIT attendees

To properly evaluate the implications of our study, it is important to clearly explain the characteristics of the group of students that are represented in our data. The students self-assessed their own level of programming experience and used this as a basis for signing up for BootIT. To help them with the evaluation, they got a description of programming concepts and were informed that *if* they were familiar with these concepts, then the course was not intended for them. Although signing up was not based on an objective assessment of their competence, we believe that self-assessment is better since it will incorporate not only *experience*, but also *confidence* and *self-efficacy*.

2.2 The goals of BootIT

The main goal of the CS0 course is to facilitate an active "hands on" learning environment wherein students learn *to* program rather than *about* programming. Along with being taught programming language constructs, students are also taught about distinctions between syntax and semantics. Apart from building the foundations of programming experience, the course also aims to boost student confidence and increase self-efficacy. The material is intentionally kept to basics with lots of opportunities for help and feedback from teaching assistants (TAs), to provide the students with successful experiences with programming, and to show them that most of their peers are at a similar level experience wise.

BootIT uses continuous alternation between short spanned (student passive) lecturing about some topic immediately followed by (student active) exercises engaging the students with the same topic, with immediate on-call assistance available from TAs.

All exercises are designed so that they conform to the Use-Modify-Create (UMC) framework [24] that organizes tasks into a competence progression hierarchy. Each topic in the BootIT course contains exercises associated with this hierarchy; e.g., exercises where the students are tasked to merely run and describe a piece of code (Use), change a piece of code so that it does something slightly different (Modify), or construct some new (simple) functionality (almost) from scratch (Create).

The exercises and examples utilised during the course all represent real-world concepts and processes. This approach has been chosen to provide the students with insight into the real-world relevance and usefulness of what they are taught.

2.3 Sample Exercise (Temperature Conversion)

For an example of this approach, let us consider a Temperature Conversion exercise for teaching basic arithmetic, use of constants, variables, and assignments. For this exercise, students are initially supplied with a code skeleton that contains everything to convert from degrees Celsius to degrees Fahrenheit except for the actual temperature conversion calculation. First, students merely run this code to observe that no conversion is performed (Use). Then, students are tasked with changing it so that it properly converts from Celsius to Fahrenheit (Modify). Finally, the students have to make a new function that converts in the opposite direction (Create).

Exercises are often surrounded by similar exercises, e.g., the temperature conversion exercise is followed by an exercise of *currency* conversion rather than *temperature*; i.e., converting between USD and DKK (using a fixed exchange rate), progressing from *known* to *unknown* examples.

Other exercises include similar practices with varying scope and ambition: Is it Friday or Not?[3] (for teaching the basics of conditional statements), a "Time Machine" that prints your age for each year in the future (for iteration), a BMI Calculator (for the interaction between multiple inputs: height *and* weight), and a Bank Account (for basic object-orientation; in particular, instantiation and manipulation of objects).

[1] One academic year is 60 ECTs (European Credit Transfer and Accumulation System).
[2] https://drive.google.com/drive/folders/1yKP14E07FINgn2izbtz6K1oZAmJFKQCP?usp=sharing
[3] Similar to http://isitfriday.org/

3 EFFECT

We consider the *effect* of the CS0 onboarding initiative through four research questions or four dimensions:

(1) **Student perspective:** How do the students perceive the onboarding initiative?
(2) **Gender diversity:** To what extent has the initiative affected gender diversity among admitted students?
(3) **Grades:** To what extent has the initiative impacted grades (competence) in the subsequent CS1 course?
(4) **Dropout:** To what extent has the initiative impacted retention during the first year?

For the last three dimensions, the effect is quantified by comparing the 271 students who attended the (CS0) onboarding initiative versus a baseline of the 527 students who did not. In total, this is based on data comprising N=798 students over a period of five years (2016–20).

3.1 Student Perspective

Through a course evaluation form, the students were invited to review the course. The evaluation form contained a total of 12 open-ended questions, of which we will detail those related to students' experiences with the course and reasons for participation. The remaining questions relate to the contents of the course and the university as a whole.

Students were invited to rate their satisfaction with the BootIT initiative on a scale from 1–7 (where, e.g., 1 was *very bad*, 4 was *average*, and 7 was *very good*). The average rating of BootIT was: 6.31. No student rated the course average or worse (1–4); more than half of the students gave it the second-highest score (6 out of 7), and almost 40% gave it the maximum score of 7.

In the student evaluation form from years 2018–2020, the students were asked to also state their *reason* for participating in BootIT (Note that these are translated from Danish to English.) By coding the open-ended qualitative parts of the answers and aggregating these, *"lacking programming experience"* was part of 79% of responses, but this number is potentially higher, as some students simply stated more vaguely that they *"wanted a good start at university."* By coding the remaining open-ended questions and comparing them, we also found that students report they sign-up to BootIT as a way to improve their competencies, suggesting the lack of confidence in their abilities; e.g., *"I hadn't programmed before and wanted some knowledge"* and *"I have tried a little coding and was unsure about my abilities."* Many students reported on insecurities about how they stacked up against their peers; e.g., *"I have learned not to be unsure about your own abilities compared to others"* and *"Meeting new people who were in as deep water as me."* After the course, students felt more prepared and calm about starting: *"I have learned a little about programming. I am more prepared to start"*, *"I feel more secure about starting. It seems less dangerous,"* and *"I am calmer about study start now!"*. All suggesting a higher degree of belief in their own ability, i.e., higher self-efficacy.

In 2021, the students (who attended BootIT, but have yet to finish their semester) were asked if they felt more ready to start their studies as a result of the onboarding course: *"Absolutely, now that I know the basics."*, *"I have never tried programming before, so I am definitely [ready] now."*, and *"Yeah, now that the first dose of stress is gone. So I definitely think it'll be easier."*

Table 1: Gender composition of the group of students who attended the CS0 course versus those who did not (2016–20).

	Women	Men	Total
CS0:	70	201	271
Non-CS0:	83	444	527
Total:	153	645	798

Table 2: Gender composition of the students on the Bachelor of Software Development: *before* CS0 was introduced (2010–15); *when* CS0 was introduced (2016); and *after* CS0 was introduced and advertized (2017–20). The 2021 data is preliminary since the semester did not end yet. (Data not included in the study is depicted in gray font.)

Year	Women	Men	Total	%Women
2010	2	69	71	3%
2011	3	66	69	4%
2012	3	63	66	5%
2013	6	64	70	9%
2014	9	84	93	10%
2015	9	80	89	10%
2016	17	126	143	**12%**
2017	33	114	147	**22%**
2018	30	119	149	**20%**
2019	27	136	163	**17%**
2020	46	150	196	**23%**
2021	35	133	168	**21%**

> **Observation 1:** BootIT attendees were generally satisfied with the onboarding initiative. Additionally, they reported that it helped them gain competence, foster confidence, and diminish insecurity.

3.2 Gender Diversity

Since its inception in 2016, CS0 has contributed to diversity by attracting more programming inexperienced students. Since women tend to self-assess as less experienced, an initiative of this kind has an indirect, but noticeable impact on diversity [7, 35]. Table 1 details the gender composition of the students attending CS0 versus those who did not. In total, approximately a *third* of all enrolled students (271 out of 798, or 34%) attended CS0. We see, however, that women are much more likely to attend CS0 than men. In fact, almost *half* of all the women enrolled (70 out of 153, or 46%) attended CS0; in contrast, less than a *third* of the men enrolled (201 out of 645, or 31%) attended CS0.

If prior programming experience were a prerequisite for the educational programme, many of the students in the top row in Table 1 would have been worse off or perhaps not even enrolled in the educational programme (Bachelor of Software Development).

Note that the motivation for opting to attend CS0 could be any combination of low self-efficacy, lack of self-perceived *competence* and/or *confidence*, but as described in Section 3.3, we can extrapolate that (almost) all the participants have no previous programming experience. Here, we, of course, need to take into account that men are more inclined, than women, to overestimate their self-perceived competence [2, 4, 12] and possibly also less willing to solicit assistance.

The increase in the number of women is also visible on the gender composition of the educational programme as detailed in Table 2 (from 2010–21). The upper part in gray font gives the numbers from *before* CS0 was instated (2010–15). We generally see a *low* percentage of women of maximum 10%. In 2016, CS0 ran for the first time. From 2017 and onwards, CS0 was adopted as a regular annual activity and its existence was heavily advertized; in particular, it was emphasized that students did not need to have any programming competencies whatsoever before starting the programme. In general, we see a much higher ratio of women (as highlighted in bold font in the table) in the programme, with a peak of 23% women in 2020. The final row in the table shows the data for students admitted in 2021, but this is not included in this study as the students did not yet finish the semester.

Please note, however, that many other diversity initiatives were also instated from 2016 and onwards (see Section 4.2), so it is difficult to isolate the effect of CS0. We therefore defensively make the following observation in relation to **RQ2**:

> **Observation 2:** Women were more likely than men to attend the CS0 initiative as it was selected by almost half of female students (as opposed to less than a third of male students).

3.3 Grades

Experience and skills prior to education have been demonstrated to be a significant predictor for first-semester programming performance [34]. Students who had no experience in programming have been shown to score 6% lower on exams and 10% lower on programming quizzes than their programming experienced peers. (As would be expected, this performance gap diminishes over time and is no longer perceptible by the end of the subsequent CS2 course.)

Table 3 shows the data on the self-reported experience level of students who attended CS0 in 2021 and those who did not.[4] We see that about half of the students who self-assessed as inexperienced (see the None column) attended CS0. Six students with Some programming experience attended CS0. Most of these students have left during the first day of CS0, as they have realized that the course was really not meant for them (in a few cases, this was also suggested by TAs). For this reason the number of students who self-assessed themselves as having a lot of experience in programming was, in fact, lower.

Table 3: CS0 attendance as a function of self-reported experience level (data from 2021).

	None	Some	Lots
CS0:	21	6	0
Non-CS0:	22	26	21
Total:	43	32	21

According to the experience-performance gap [34], students in the None column ought to perform *worse* (than average), the ones in the Some column should perform about *average*, and the ones in the Lots column should perform *better* (than average). If BootIT did *not* have any effect, the cohort of (mainly inexperienced) students who attended the CS0 course would predictably perform worse than the cohort of students (with mixed experience levels) who did not attend it. In other words, the average CS1 grades of BootIT students should be *worse* than those of the Non-CS0 students. The reason for it is simply that the set of non-attendees contained more experienced students than the set of attendees.

If, however, BootIT "works" as intended, the (minimal) effect of it would be that the inexperienced students would be effectively *moved* from the category None to the category Some (now having *some* programming experience). Based on this student experience distribution, one would predict little to no differences in the average grades of the two cohorts (BootIT vs Non-CS0 students). It is so, because after the course both groups consist mostly of people with "some" experience in programming.

This is exactly what we see: Figure 1 shows the distribution of the first-attempt grades for the CS1 (Introductory Programming) course for those who attended CS0 versus a baseline of those who did not. The grade distribution among the two groups is almost identical which is also evident by the averages of pass grades[5] which are given in Table 4. (Data from 2021 is not included since we do not yet have all the data.[6]) The average of the pass grades among students who attended CS0 was 6.89 versus 7.01 for the ones who did not attend. The difference in average is only about a tenth of a grade point (0.12).[7] Table 4 also shows that, curiously, the percentage of students receiving the *highest* grade (A) is, in fact, slightly higher among students attending CS0 (7.6%) compared to the ones who did not take the onboarding course (only 7.1%), although the difference is *not* statistically significant (p = 0.79).

[4] Students enrolled in the CS1 course of 2021 were asked to fill out a questionnaire of a single question detailing whether they had no experience (HTML and Excel was explicated as not counting), some experience (10-250 lines of self-written coherent code), or lots of experience (ranging from more than 250 lines of code to professional developer). The respondents were also explicitly asked not to include experience from the CS0 course. In total, 96 students out of the 186 students answered.

[5] In Denmark, it is customary to compute the grade average of a *course* using the pass grades only (the average for a large population ought to be around 7).

[6] The 2021 data will be included as soon as we have it.

[7] Note that the numerical distance between two adjacent grades in the Danish grade scale is *two* or *three*

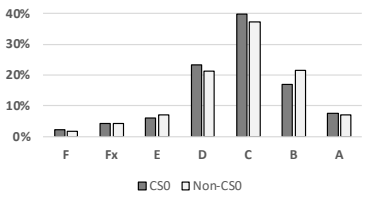

Figure 1: Distribution of CS1 grades for those who attended CS0 versus those who did not (2016–20). The Danish grade values (-3, 00, 02, 4, 7, 10, 12) are shown along with the corresponding ECTS grade-scale letters (F, Fx, E, D, C, B, A). F & Fx are *fail* grades whereas E–A are *pass* grades.

Table 4: Grade statistics for the CS1 course for students who attended CS0 versus those who did not (2016–20).

	CS0	Non-CS0
Pass grade average:	6.89	7.01
Highest grade (A):	7.6%	7.1%
Failed course (F or Fx):	6.4%	6.0%

Similarly at the other end of the grade spectrum, the number of students *failing* the exam (grades F or Fx) is virtually the same among the two groups (6.4% vs 6.0%). Unsurprisingly, the difference between the grades among the two groups is *not* statistically significant ($p = 0.55$) using a Mann-Whitney U non-parametric test[8] which does not assume that the data is normally distributed.

Identical to our findings, Aarhus University (AU) who also employ an onboarding initiative for inexperienced students on their Bachelor of Computer Science finds that attendees (without prior experience) are also 0.1 CS1 grade points behind students with prior experience.[9] Interestingly, AU additionally has a group of "little experienced" students who are apparently *not* offered their (distributed) onboarding; those students are 0.4 CS1 grade points behind which is an indication of a positive effect of onboarding courses, in general. Comparingly, early onboarding has the advantage over distributed semester-long onboarding that the students will be more productive from their first day of CS1.

> **Observation 3:** Despite the lack of confidence in their experience level before starting at university, CS0 attendees were able to achieve the same level of competence as students who evaluated themselves highly.

3.4 Dropout

Table 5 shows the dropout rates (inverted retention) *during* the first and second semester for those who attended CS0 versus a baseline of those who did not. This is based on the students that initiated their studies in 2016–20. We show only first-year dropout as we find it unlikely that later dropout would be attributable to lacking programming experience and/or confidence before starting university, years earlier. The percentage of students dropping out during the first semester is coincidentally exactly the same among the two groups: 6.3% (in both cases). The dropout rate during the second semester is 8.9% among the students that attended CS0 and only slightly lower (7.4%) among those who did not.

Table 5: Dropout rates during the first year for those who attended CS0 versus those who did not (2016–20).

Dropout:	CS0	Non-CS0
1st Semester:	6.3% (17/271)	6.3% (33/527)
2nd Semester:	8.9% (24/271)	7.4% (39/527)
1st Year:	15% (41/271)	14% (72/527)

We use a standard Z-score test[10] to check if one proportion is statistically significantly larger than the other. The dropout ratios are *not* statistically significantly different with $p = 0.99$ and $p = 0.47$ for first and second semester dropout, respectively.

> **Observation 4:** CS0 attendees were *not* dropping out more frequently than their peers who did not sign up for the onboarding course.

4 THREATS TO VALIDITY

We first consider *construct validity* and scrutinize how the data and metrics involved were obtained and measured. Then, we examine the *internal validity* and the threats to the validity of the study itself and thus to its findings. Finally, we treat the *external validity* and ponder the generalizability of our findings.

4.1 Construct Validity

Measuring gender? The information about the gender of participants was automatically obtained from the enrollment system based on information from the Danish central person registry (CPR). The gender in the registry is binary, but people can have their gender information changed to reflect their self-identification. Clearly, it would have been better to have had the students self-identify their gender, with choices beyond the binary dichotomy of *female* xor *male*. In retrospect, we did not ask the students for this information during CS0 as we did not anticipate that the data would later be used in connection with a gender-related analysis.

Measuring CS0 attendance? Our data is based on students *signing up* for rather than *attending* CS0 since this was what had been recorded by the university. There may, of course, be students who signed up without attending (e.g., due to illness), but we expect this to be an insignificant number, not affecting the statistical analyses. In contrast, students not signed up would not have been able to attend. The attendance at CS0 was generally comparable to what had been announced by the ITU study administration which fit the capacity of the teaching venues. (In 2020, more space was allocated due to the Covid-19 pandemic.)

Measuring dropout and grades? The sensitive dropout and grade data were obtained by the ITU Statistics Unit according to

[8] https://www.socscistatistics.com/tests/mannwhitney/
[9] Data from Kaj Grønbæk, Head of CS Department, Aarhus University.
[10] https://www.socscistatistics.com/tests/ztest/

their standard procedures. The aggregation of this sensitive information with CS0 attendance was also conducted by the ITU Statistics Unit. For this paper, we were provided with only aggregated anonymized data that does not reveal any information about individual students.

4.2 Internal Validity

Can we equate CS0 attendees with "students who lack experience"? Because of the way CS0 is organized and, especially, advertised, it is tempting to treat the division between students who attended the course and those who did not as a proxy for programming (in)experience. Unfortunately, this cannot be done for two reasons. First, we cannot be sure of the reasons why students who did not attend CS0 decided not to enroll in the course. It is possible that they did perceive themselves as inexperienced, yet did not sign up for unrelated reasons. Second, students chose to enroll based on their own self-assessment. This indicates the lack of confidence but not necessarily the lack of experience. We tried to mitigate it by giving the students a quick method of evaluation of their experience (concept recognition described in section 2.2) but we did not test their competence level directly.

Were the two groups really that different to begin with? One may object that the two groups (attended vs unattended) fared similarly, simply because there never were any significant differences between the two groups, to begin with. However, as succinctly stated in [1]: *"Previous work has shown that inexperienced students under-perform their experienced peers when placed in the same introductory courses, and are more likely to drop out of the CS program."* Besides, the teachers and TAs perceived that the students did *not* have basic programming competencies on the first day of CS0, in mid August; and that the students *did* have basic programming competencies on the first days of CS1, in late August. As mentioned earlier, the risk of incorrect self-assessment has also been decreased by the fact the students had the ability to quickly compare their knowledge to the level assumed in CS0.

Can the diversity increase be attributed to CS0? From 2016 and onwards, ITU introduced several *other* diversity initiatives, many of which were deliberately aimed at recruiting women (e.g., IT-Camp and Coding Café). Many of these initiatives involved showing female high-school students: (1) what programming is; (2) that they fit in; (3) that they can learn how to program; *and* (4) that programming can be used to address interesting real-world problems. CS0, however, is different in serving specifically to *widen* the recruitment profile to also include applicants (male and female) who have never programmed before and to ease the beginning of the Software Development education. It is an initiative that complements well with many other diversity initiatives. We cannot isolate the effect of CS0 from that of other diversity initiatives, but we can say that CS0 facilitates the recruitment of students without prior programming experience. Our assumption of the influence of CS0 on gender diversity has been based on the correlation between self-assessed inexperience and gender as well as on the fact that, as suggested by this correlation, the percentage of women attending in the course was higher than that of the men.

Can the equivalent dropout and grades be attributed to CS0? Both the first-year dropout and CS1-grades were similar among the group of students who attended vs those who did not attend CS0. One may object that the students just "caught up" somehow (without it being due to CS0). Also, there were a number of initiatives aimed at helping inexperienced students *during* the semester (e.g., Live Coding and a Study Lab) instead of *before* the semester (as was the case of CS0). We still believe that CS0 played a significant role in students catching up as they were able to write non-trivial programs on the last day of CS0. The results from AU's distributed semester-long onboarding (see Section 3.3) provides an indication: their onboarded students (without programming experience) caught up, whereas their un-onboarded students (with little experience) did not. This is an indication that the cause was, in fact, onboarding. There is, however, a way to isolate experimentally the effect of CS0 from the other interventions: simply offer CS0 randomly to only half of the inexperienced students and subsequently compare those that attended versus those who did not. This experiment, however, comes with non-trivial ethical considerations.

Is there a connection between onboarding courses and the increase in gender diversity? As we pointed out in the introduction the connection between gender and the onboarding initiatives such as CS0 remains to be indirect as it exists due to the existence of another correlation - that of gender and self-estimated lack of programming experience. As can be seen in Observation 3, this connection can be seen in the case of our case study. The lack of direct connection makes it difficult to measure the influence of onboarding initiatives on the increase of gender diversity. It is even less clear because of the existence of additional factors, such as other initiatives aimed at increasing diversity. Still, we claim that the connection between gender and the perception of the level of experience—the sheer fact women tend to be less confident in their pre-study competence shows that any action aimed at students with low confidence has an indirect impact on gender diversity.

Teacher and TA involvement in this study. Since a couple of people were involved in both teaching CS0 (as teacher or TA) *and* writing this paper, there is a risk of (inadvertent) observer/experimenter effect bias. It is nonetheless important to point out that all of the activities related to this paper (down to the formulation of hypotheses and research questions) happened long after the course had ended. To our defense, we never knew we would conduct this study until we recently found out that the data on who attended CS0 had, in fact, accidentally, not been deleted. The paper is thus essentially based on "historically" collected data.

Impact of the Covid-19 pandemic. On March 13, 2020, Denmark entered a national lock-down in response to the Covid-19 pandemic.[11] All physical teaching was suspended and transferred online. By mid-August, however, CS0 managed to run physically (albeit with Covid-19 distancing requirements). The subsequent CS1 course, however, ran virtually from late August 2020. This obviously impacts this study. Nonetheless, we have not excluded "the Corona year" (2020) from this study as the 2020 data does not exhibit any different pattern than the data from the other years. All observations and conclusions would have been the same if we had excluded 2020 and based the study on data from 2016–19 only.

[11] ITU responded swiftly by locking down already two days earlier.

4.3 External Validity

Beyond Software Development? This study was carried out in the context of a *Software Development* educational programme. Since there is nothing intrinsic to Software Development in this study aside from the role of programming, we expect this generalizes to any (Computing) education for which programming plays a fundamental role in the educational programme; in particular, Computer Science and Software Engineering.

Beyond ITU and Denmark? Our study took place in a Danish educational context with predominantly Danish students. We expect the results to generalize to other countries; especially ones with similar access to computers and values such as a focus on individuality, autonomy, and responsibility for own learning.

5 RELATED WORK

While CS0 is not unique as similar initiatives are implemented at many other universities, it is interesting that, to the best of our knowledge, no paper quantifies the *effects* of such a short spanned pre-university onboarding initiative. There appears to be related work considering either *lifting inexperienced students* or *increasing student diversity*, but not both at the same time (as in our study).

Lifting Inexperienced Students. The documented impact prior programming inexperience has on performance during CS1 [9, 18, 22] prompts the question: How do we aid the less experienced? To answer this, studies often examine either *differentiated* CS1 courses or *fallback* CS0 courses.

Some universities offer multiple versions of CS1 courses, catering to both those *with* experience as well as to those *without* [11], the latter requiring an initial eligibility assessment. Other universities assess students initially and *strongly encourage* students deemed insufficiently competent to voluntarily register for the fallback CS0 course rather than directly for CS1 [13]. Other universities simply "force" (involuntary) enrollment based on such tests [26].

Furthermore, studies also explore different approaches to CS0 as a way of improving performance and retention within CS1 [28, 30] and different methods and practices which have proved valuable in improving the inclusion of novice programmers [5, 15].

Increasing Student Diversity. Increasing the diversity among the computing student body has been a priority for many universities, especially over the last decade. A large focus has been on gender imbalance, which has led to a number of interventions to involve more women and share these interventions among institutions [33]. This includes initiatives in CS1 such as; considering diversity in incitements [14], displaying the relevance to various types of students [31], and increasing materials' relation to both reality [16] and people [10, 25]. All aiding in creating an inclusive environment [32]. This CS0 attempts to embrace all of this, to create a more inclusive environment [6].

6 CONCLUSION

As pointed out in the introduction, the lack of initial programming experience can often have a lasting effect on the education process of IT students. As argued in [1], educators should not be blinded by positive cases of students who were able to surmount the challenges and easily catch up with their experienced peers as they are rather an exception than the norm. Regardless of how effectively the students are able to make up for the initial discrepancy, it can be argued that ironing out the differences between them is the safest way of minimizing dropout rates and closing the CS1 grade gap. The reason for this is that while students are statistically able to catch up, this is irrelevant from the psychological perspective. A student struggling with the material and course does not know early on that (s)he will eventually catch up and thus may get discouraged prematurely. Thus, any attempt to increase self-efficacy of students should be implemented in an attempt to decrease drop-out rates. This is particularly interesting in the light of the correlation between gender and prior programming (in)experience. Since women are more likely to be inexperienced, the difficulty with catching up contributes to the well-known problem of gender imbalance in IT. This paper presented a closer look at the effects of an elective onboarding initiative offered to the students at a Danish university. We argue that an initiative of this type can positively affect three aspects of early education that are related to the problem; it may serve to:

(1) increase the **competence** of participants;
(2) increase the **confidence** of participants; *and*
(3) decrease the **insecurity** of participants.

Due to the correlation between perceived inexperience and gender, any action that helps inexperienced students plays a double role as it additionally helps to solve the problem of lack of gender equity. We argue that running an onboarding course is an effective mean that is advisable even in the case of short, condensed (three days) courses similar to the CS0 course introduced here. Thus, we provide the material used to teach the course on this link.[12]
Since the length of the course should not have any bearing on the second and the third effect, deciding on a condensed version may be even preferable because of cost-effectiveness and the lack of risk of an overlap with existing introductory courses. As a result, we recommend the following:

> **Recommendation:** We recommend that universities consider instating pre-university onboarding initiatives as a preemptive measure for potentially decreasing dropout, for closing the prior experience and confidence grade gap, and for potentially increasing student diversity.

ACKNOWLEDGMENTS

The authors thank Rolf Fagerberg for sharing experiences with an onboarding course at the University of Southern Denmark (SDU) and inspiring us to study our data. We thank Kaj Grønbæk for sharing the effects of a distributed onboarding initiative at Aarhus University (AU). We also thank course programme administrator Allette Bjørn Bundgaard for uncovering data on who attended the five CS0 editions and Jane Andersen from the ITU Statistics Unit for joining this data against the student data and providing it to us in anonymized form. We thank Melissa Høegh Marcher for valuable feedback and suggestions. Finally, we thank the many CS0 TAs throughout the years for their assistance with running the course.

[12] https://drive.google.com/drive/folders/1yKP14E07FINgn2izbtz6K1oZAmJFKQCP?usp=sharing

REFERENCES

[1] Christine Alvarado, Gustavo Umbelino, and Mia Minned. 2018. The Persistent Effect of Pre-College Computing Experience on College CS Course Grades. *SIGCSE '18: Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 876–881.

[2] Brad M Barber and Terrance Odean. 2001. Boys will be boys: Gender, overconfidence, and common stock investment. *The quarterly journal of economics* 116, 1 (2001), 261–292.

[3] David John Barnes, Michael Kölling, and James Gosling. 2006. *Objects First with Java: A practical introduction using BlueJ.* Pearson Prentice Hall London.

[4] Claes Bengtsson, Mats Persson, and Peter Willenhag. 2005. Gender and overconfidence. *Economics letters* 86, 2 (2005), 199–203.

[5] Cathy Bishop-Clark, Jill Courte, and Elizabeth V. Howard. 2006. Programming in Pairs with Alice to Improve Confidence, Enjoyment, and Achievement. *Journal of Educational Computing Research* 34 (3 2006). Issue 2. https://doi.org/10.2190/CFKF-UGGC-JG1Q-7T40

[6] Valeria Borsotti. 2018. SIGSOFT Distinguished Paper - Barriers to Gender Diversity in Software Development Education: Actionable Insights from a Danish Case Study. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* 146–152.

[7] T. Busch. 1995. Gender differences in self-efficacy and attitudes toward computers. *Journal of educational computing research* 12, 2 (1995), 147–158.

[8] Pat Byrne and Gerry Lyons. 2001. The effect of student attributes on success in programming. *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education* (2001), 49–52. https://doi.org/10.1145/377435.377467

[9] Pat Byrne and Gerry Lyons. 2001. The effect of student attributes on success in programming. In *Proceedings of the 6th annual conference on Innovation and technology in computer science education.* 49–52.

[10] Ingrid Maria Christensen, Melissa Høegh Marcher, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving PEOPLE Rather Than THINGS Appeal More to Women (Recruitment Perspective). *ACM International Computing Education Research conference (ICER 2021)* (2021).

[11] James P Cohoon and Luther A Tychonievich. 2011. Analysis of a CS1 approach for attracting diverse and inexperienced students to computing majors. In *Proceedings of the 42nd ACM technical symposium on Computer science education.* 165–170.

[12] Shelley J Correll. 2001. Gender and the career choice process: The role of biased self-assessments. *American journal of Sociology* 106, 6 (2001), 1691–1730.

[13] Charles Dierbach, Blair Taylor, Harry Zhou, and Iliana Zimand. 2005. Experiences with a CS0 course targeted for CS1 success. *ACM SIGCSE Bulletin* 37, 1 (2005), 317–320.

[14] Allan Fisher and Jane Margolis. 2002. Unlocking the Clubhouse: The Carnegie Mellon Experience. *SIGCSE Bull.* 34, 2 (June 2002), 79–83. https://doi.org/10.1145/543812.543836

[15] T. Flowers, C.A. Carver, and J. Jackson. 2004. Empowering students and building confidence in novice programmers through Gauntlet. In *34th Annual Frontiers in Education, 2004. FIE 2004.* T3H/10–T3H/13 Vol. 1. https://doi.org/10.1109/FIE.2004.1408551

[16] Mark Guzdial. 2003. A Media Computation Course for Non-Majors. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education* (Thessaloniki, Greece) *(ITiCSE '03).* Association for Computing Machinery, New York, NY, USA, 104–108. https://doi.org/10.1145/961511.961542

[17] Dianne Hagan and Selby Markham. 2000. Does it help to have some programming experience before beginning a computing degree program? *ITiCSE '00: Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education* (2000), 25–28. https://doi.org/10.1145/343048.343063

[18] Dianne Hagan and Selby Markham. 2000. Does it help to have some programming experience before beginning a computing degree program?. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education.* 25–28.

[19] Edward Holden and Elissa Weeden. 2003. The Impact of Prior Experience in an Information Technology Programming Course Sequence. *CITC4 '03* (2003), 41–46.

[20] Edward Holden and Elissa. Weeden. 2003. The impact of prior experience in an information technology programming course sequence. *CITC4 '03: Proceedings of the 4th conference on Information technology curriculum* (2003), 41–46. https://doi.org/10.1145/947121.947131

[21] Edward Holden and Elissa. Weeden. 2004. The experience factor in early programming education. *CITC5 '04: Proceedings of the 5th conference on Information technology education* (2004), 211–218. https://doi.org/10.1145/1029533.1029585

[22] Edward Holden and Elissa Weeden. 2004. The experience factor in early programming education. In *Proceedings of the 5th conference on Information technology education.* 211–218.

[23] Päivi Kinnunen and Beth Simon. [n.d.]. CS Majors' Self-Efficacy Perceptions in CS1: Results in Light of Social Cognitive Theory. ([n. d.]), 8.

[24] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational Thinking for Youth in Practice. *ACM Inroads* 2, 1 (Feb. 2011), 32–37. https://doi.org/10.1145/1929887.1929902

[25] Melissa Høegh Marcher, Ingrid Maria Christensen, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving PEOPLE Rather Than THINGS Appeal More to Women (CS1 Appeal Perspective). *ACM International Computing Education Research conference (ICER 2021)* (2021).

[26] C. Marling and D. Juedes. 2016. Cs0 for computer science majors at ohio university. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (2016), 138–143.

[27] Allison Master, Sapna Cheryan, Adriana Moscatelli, and Andrew N Meltzoff. 2017. Programming experience promotes higher STEM motivation among first-grade girls. *Journal of experimental child psychology* 160 (2017), 92–106.

[28] Kris Powers, Stacey Ecott, and Leanne M Hirshfield. 2007. Through the looking glass: teaching CS0 with Alice. In *Proceedings of the 38th SIGCSE technical symposium on Computer Science Education.* 213–217.

[29] Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education.* 171–175.

[30] Mona Rizvi, Thorna Humphries, Debra Major, Meghan Jones, and Heather Lauzun. 2011. A CS0 course using Scratch. *Journal of Computing Sciences in Colleges* 26, 3 (2011), 19–27.

[31] Eric S. Roberts, Marina Kassianidou, and Lilly Irani. 2002. Encouraging Women in Computer Science. *SIGCSE Bull.* 34, 2 (June 2002), 84–88. https://doi.org/10.1145/543812.543837

[32] Roli Varma. 2006. Making Computer Science Minority-Friendly. *Commun. ACM* 49, 2 (Feb. 2006), 129–134. https://doi.org/10.1145/1113034.1113041

[33] Telle Whitney and Valerie Taylor. 2018. Increasing Women and Underrepresented Minorities in Computing: The Landscape and What You Can Do. *Computer* 51, 10 (2018), 24–31. https://doi.org/10.1109/MC.2018.3971359

[34] Chris Wilcox and Albert Lionelle. 2018. Quantifying the benefits of prior programming experience in an introductory computer science course. In *Proceedings of the 49th acm technical symposium on computer science education.* 80–85.

[35] E. B. Witherspoon, C. D. Schunn, R. M. Higashi, and E. C. Baehr. 2016. Gender, interest, and prior experience shape opportunities to learn programming in robotics competitions. *International Journal of STEM Education* 3, 1 (2016), 1–12.