Why do CS1 students become repeaters?

María-Jesús Marco-Galindo, Julià Minguillón, David García-Solórzano, y Teresa Sancho-Vinuesa

Abstract—**Learning to program is hard for many students. As a result, CS1 courses have a significant percentage of repeaters. For this reason, the goal of this article is to analyze which factors affect repeaters so that a specific learning strategy for them can be performed. In this regard, a first analysis of a CS1 course shows there are two types of repeaters: (1) those who do (almost) nothing throughout the semester and drop out, and (2) those who work during the whole semester, but finally fail. According to repeaters' perceptions, they were motivated to learn to program, but it was difficult for them to keep up because of it was so hard to reconcile it with their personal context, so based on what they did the previous semester, they would prefer to continue from where they left off or change the pace of activities.**

Index Terms— **CS1 dropout factors, introductory programming course, performance analysis, repeaters' perception**

I. Introduction

SEVERAL articles –such as [1]– state that a lot of students are still challenged by programming, because of the amount of difficulties that they have to overcome in order to master and acquire the contents and competencies of an introductory programming course (aka CS1). This is reflected in the academic results of CS1, whose behavior is almost bimodal [2], with a group of students that progresses adequately achieving good results, and another htoup formed by students who fail after making, for the most part, a great effort during the semester. As a consequence, the success rates of CS1 courses are low [3] and the dropout rates are around 28-33% [4]. Given this scenario, it is easy to understand why the percentage of repeaters in initial programming subjects is not negligible and deserves proper attention.

In our educational system is not possible to avoid having repeater students in introductory programming courses. For this reason, the main objective of this research is to gain an insight into the difficulties experienced by those students who did not pass the subject, in order to design a teaching intervention that helps them pass the course when they retake it. Specifically, this article is structured as follows: Section II presents a brief state of the art on the factors that influence dropout in introductory programming courses. Section III also compiles the main characteristics that determine, according to the literature, the profile of students who retake CS1. Section IV poses the questions of the present research and explains the methodology used to answer them. The results are shown in Section IV, which are discussed in Section VI. The article concludes in Section VII which also discusses the future work.

This article is an extension of a previous research work [5] that was awarded best paper in the JENUI 2021 Conference. With regard to the previous paper, the present article expands the literature review and adds a third research question about the perception of the students repeating. To answer it, a questionnaire was designed and sent. Its results were analyzed and discussed according to the goal of this research.

Manuscript received XXXX; revised December XXXX; accepted XXXX. Date of publication XXXX; date of current version XXXX. (Spanish version received XXXX; revised XXXX; accepted XXXX). This work was supported in part by the Catalan Government under Project 2017SGR1619. (Corresponding author: María-Jesús Marco-Galindo.)

María-Jesús Marco-Galindo, Julià Minguillón, David García-Solórzano and Teresa Sancho-Vinuesa are with the Computer Science, Multimedia and Telecommunication School, Universitat Oberta de Catalunya, 08018 Barcelona, Spain (email: mmarcog@uoc.edu; jminguillona@uoc.edu; dgarciaso@uoc.edu; tsancho@uoc.edu).

DOI (Digital Object Identifier) Pending

II. Dropout Reasons

This section identifies the factors that, according to the literature, favor dropout CS1 courses. Such factors are sorted by following the three categories defined by Lee and Choi [6]: course/program factors, environmental factors, and student factors. Thus, the first factor depends on the design of the course, and the latter, on the student and her circumstances.

A. *Concepts are Tightly Integrated*

One of the difficulties presented by CS1 courses is, as Robins [7] points out, the fact that the concepts are tightly integrated, so that the elements introduced later depend on those acquired earlier. This fact, together with the complexity of the teaching approach of the subject itself [8], means that many students try to advance without correctly assimilating all the concepts and skills, until reaching a point where they have accumulated so many deficiencies that they are in a situation of collapse and, consequently, giving up is their only choice. Additionally, as [9] states, the concepts initially presented tend to be used again and re-evaluated indirectly throughout the course. In this regard, it should be noted that even methodologies that seek to prevent a concept from being learned without mastering a previous one, such as mastery learning, can lead students to procrastination [10].

B. *Programming Language Used in the Course*

The goal of any introductory programming course should be learning to program, not learning C, for example. This implies that attention should be paid to problem solving as well as algorithm understanding and design. This also means that the programming language used during the course must be a mere vehicle for this purpose, that is, it should be treated as another teaching resource. However, the reality is that neither programming languages are designed to learn programming nor do students know how to differentiate between what programming is and what a programming language is [11]. Thus, there are students who fail or drop out, not because of the instructional design of the course, but due to the complexity of the programming language used [12] and the difficulties inherent in the implementation of algorithms.

C. Types of Activities

Some training activities –such as a multiple choice test– can give students the wrong perception of their degree of acquisition of concepts and skills [13]. In the same way, some activities carried out in pairs may imply that the student, without being aware of it, has the perception of being capable of finishing it individually, when the reality is that without the help of her partner, she would not have finished it. Therefore, students can have a false perception of the real knowledge that they have of the course, of which they are not aware until they carry out an individual and assessable assignment.

D. Course Schedule

Research works such as [14] show that, in general, students may need more time than the teacher estimates to acquire the programming knowledge expected in the course. Likewise, some works [15] state that, often, at the end of the semester, the pace speeds up, being too fast for many students. Perhaps one of the reasons for this acceleration is due to the fact that, as Luxton-Reilly [8] points out, teachers' expectations are unrealistic and, therefore, too many new concepts are introduced at the end of the course, assuming that the fundamentals have been assimilated by the students.

E. Adaptation to the University

As Lowe and Cook [16] indicate, a not inconsiderable number of students have difficulties in bridging the gap between school and university due to various factors. For example, according to such study, around 40% felt that the teacher-student interaction was less helpful and friendly than in previous levels of study, which was harmful to students' chances of success. Thus, some learners who drop out of CS1 courses might have actually dropped out of the university.

F. Ineffective Study Strategies

Universities require students to be more autonomous in comparison with lower levels of education. However, many students are not capable of having become self-regulated learners. According to [16], almost a third of students had absolute difficulties in carrying out self-regulated learning.

As far as the introductory programming courses themselves are concerned, Petersen et al. [13] states that a lot of students perceive that the study strategies they use in other courses –more focused on reading lecture notes– do not work for programming courses. According to these authors, this fact affects the low performance and even dropout of students. In this regard, works such as [11] also point out the difference between the studying approach used in other subjects (even at school) and the one that is necessary to learn programming. Likewise, a comparison [17] between students with high and low performance identified that the latter tend to memorize specific code solutions, instead of understanding the underlying concepts. Similarly, Hawi [18] emphasizes that the learning strategy was the main reason of failure. In fact, he indicates that many students realize that there is a strategy, based on continuous practice, which increases the chance of passing the course.

G. Lack of Time

Some students find that programming assignments take too much time and they prefer to concentrate on other courses [13], [19], [20]. In fact, many students do not know how to perform the exercises, so they get stuck spending a lot of time and give up [19].

Other students manage their time poorly and start the assignments too late, so they do not have enough time to finish activities before the due date [19].

Finally, there are personal and/or professional situations of the student that may lead to a reduction in the time available for the subject and force them to dropout.

H. Lack of Confidence when Programming

Some investigations such as [21] have seen that those students that believe in a fixed programming-aptitude can tend to drop out. Besides, students often evaluate their own progress, which can give them a false sense of their ability to program, affecting directly their self-efficacy [22]. During these self-assessments, students take into account elements such as their own expectations, the time needed to carry out the activities, the comparison with others, etc. Such factors can even make them believe that they understand a concept, when in fact they do not understand it or do not master it sufficiently [23].

I. Lack of Motivation

Motivation, along with confidence, is a very relevant affective factor in any learning process. In this regard, many students are frustrated when they see that they do not progress despite dedicating time and effort to the course [13], [19].

There are also students who feel isolated because they do not receive enough support from their instructor, or such help is given too late [13]. This requires students to have a high intrinsic motivation that, when absent, can frustrate them and lead them to give up.

Likewise, non-computer science students do not find a relationship with their degree, especially in terms of examples and exercises [24]. Contextualizing exercises makes students perceive programming concepts as more relevant and, moreover, contributes to higher success rates [25].

III. Repeating Students' Profile

There are very few studies that analyze the behavior of repeating students in the field of programming. One of them is that of Sheard and Hagan [26], who point out that many repeaters do not have a great motivation for programming, not even for computer science, but that they chose such degree, among other reasons, for having a best career opportunity or for not having achieved the necessary access grade to do the desired degree. The same research shows that the most of repeaters do not want jobs directly related to programming.

Another differentiating factor, analyzed for example in [27], is self-esteem, which is lower in repeaters due to having failed the courses in a previous semester.

On the other hand, the activities in the first weeks of the course can be perceived by repeaters as a waste of time. This is demonstrated by the results obtained by [28] through a monitoring tool, where the participation in the initial exercises was lower by the repeaters compared to the students who were taking the subject for the first time. However, according to [29], pair programming activities in the initial weeks increases the motivation of the repeaters. One of the possible reasons is the fact of being able to discuss different solutions for the same exercise with another student of a similar level. In the same vein, the research conducted by Sheard and Hagan [26] also emphasizes that the usefulness of team work was greater for repeating students than for new ones.

IV. Methodology

As part of a broader investigation on the design and analysis of interventions in CS1 courses [30], this paper aims to answer the following research questions:

- RQ1: Are there differences in the outcomes of repeating students depending on what they did in the previous run?
- RQ2: Are there traits in repeaters' profiles that might explain such differences in their results?
- RQ3: Which reasons do the repeating students think that led them not to pass the course?

Based on the analysis of the results of repeating students and a literature review, the factors that may be the reason for not passing at the first attempt were identified. Afterward, a survey was designed to know the repeaters' perception of the influence of such factors in their failure.

A. Context

The "Fundamentals of Programming" course is compulsory in the Bachelor's degrees in Computer Engineering and Telecommunications Technology Engineering at the Universitat Oberta de Catalunya (UOC), but it is also a formative complement in some specialized Master's degrees, as well as taken as an elective course in other degrees. The profile of the students is very heterogeneous, as the UOC is a fully online university aimed at adult students. This course was redesigned in the 2017/18 academic year with the aim of helping students to acquire skills and competencies progressively by using formative assessment [31]. As an introductory course, the basic principles of programming are introduced from algorithmics and combined with the practice of simple exercises in C language, which progressively get more difficult. Continuous assessment is used, and it is based on a sequence of assignments that combine algorithmic design and programming exercises in C language. The assignments are optional (i.e., it is not compulsory to do them all), so the student decides how many and which ones to do, taking into account that each of them represents a part of the final grade of the continuous assessment. Students have different educational resources to do the assignments correctly: theoretical contents of algorithms, indications and examples of coding in C, and a virtual machine with the Codelite IDE already installed. Each week an assignment is submitted and then its solution, as a group feedback, is published. Once a set of four assignments is completed, the grade for each of them is received, as well as personalized feedback from the teacher. The teacher uses the virtual classroom board to communicate any issue related to the assignments. Nevertheless, student's doubts are shared and answered through the virtual discussion forums or the teacher's email. On discussion forums, students are expected to participate actively, thus creating knowledge collaboratively. In addition, they have a specific classroom, called "C Laboratory", where doubts about the environment and the programming language are answered.

B. Exploratory Analysis

The data for the analysis was collected from the institutional Learning Record Store (LRS) [32], which stores all the evidence related to the activities carried out by the student, from enrollment to the final grades obtained in the subjects, including assignments' grades. The study includes all those students, whether repeaters or not, who have enrolled in the "Fundamentals of Programming" course at least once from the 2017/2 to the 2019/1 semester (i.e. four semesters), which represents a total of 1,206 students. Therefore, this study was carried out with the whole population of interest. Each semester, the students are different and the different cohorts (semesters) are not completely equivalent, but considering that the course design is the same since the 2017/18 second semester (aka 2017/2), the analysis was performed for all students in a single dataset, taking into account their results from the previous semester. The following data is available for each student:

- Gender: Male (976, 80.9%); Female (230, 19.1%).
- Age range: E1 - until 20 years old (41, 3.4%); E2 - from 21 to 30 (538, 44.6 %); E3 - from 31 to 40 (353, 29.3 %); and E4 - 41 years old or over (274, 22.7 %).
- Are they taking the course as part of the Bachelor's degree in Computer Science? YES (1,016, 84.2%); NO (190, 15.8%).

Moreover, for each student and semester, the following data related to their academic activity is also available:

- Number of courses enrolled and passed at the same time (not including "Fundamentals of Programming").
- Grades obtained for each weekly assignment.

C. Measurement Instrument: Questionnaire

As a task prior to the design of the questionnaire, based on the literature and the aforementioned categorization of Lee and Choi [6], the elements of the course on which action can be taken were detected, as well as other factors related to the stu-

TABLE I
DETERMINANT FACTORS FOR THE DESIGN OF AN INTERVENTION

Factor	Description	Type
Motivation	Students do not look for additional information or face the challenges presented to them [17], [19], [33].	1
Beliefs and attitude	Students assume that dropping out a programming course is normal, or that learning programming is very difficult [13], [34]	1
Time management	Students lack self-regulation and planning skills [17], [19].	1
Study strategy	Students use inappropriate study techniques for programming (e.g. memorization) [13], [17].	1
Difficulty	The contents of the course are complex and/or cover too many concepts [8], [35], [36].	2
Theoretical resources	Learning resources are not clear, sufficient and/or well organized and up to date [2].	2
Programming language and environment	The programming language is not appropriate to learn and/or the programming environment is not easy to install and use [37].	2
Type of assignments	There is not enough variety of assignments and these are not contextualized or help to learn programming progressively [13], [19].	2
Assignments design	The statements of the assignments are not clear and understandable, the resources and materials available are not sufficient to solve them and/or the workload is very high [38].	2
Assignment schedule	The number and pace of the assignments are not adequate [13].	2
Teacher support	Teacher's support and feedback is inadequate and/or late [13], [19].	2
Social presence	The communication channels with teachers and between peers are not adequate to favor a sense of belonging to the group [39].	2
Family reconciliation	Incompatibility with work and/or family responsibilities [40].	3
Unexpected events	Unforeseen circumstances such as illness or job issues [13].	3
Learning environment	Problems of availability and accessibility to the virtual campus in the case of online studies and/or difficulties in dealing with the environment [40].	3

dent. Table I summarizes such items, emphasizing those works in which they are described and their typology according to the aforesaid classification. Socio-demographic factors of the student or others related to her academic situation, which can be extracted from the LRS, have not been included.

Each of the factors described in Table I was integrated as an item in a questionnaire, which aim was to know the repeating students' perception of the reasons why they failed CS1. In this way, it could be known, for example, if the current feedback is perceived differently by students who fail or drop out, and, therefore, it is necessary for the teacher to act differently according to the student's profile. The questionnaire consists of 15 questions divided into five categories: (1) difficulty and interest, (2) resources, (3) activities, (4) communication and (5) personal context of the student (see Table II).

In addition to the 15 questions based on five-level Likert scale, a final question asks students for ranking each of the following phrases in terms of its importance:

- "I wish I could continue from the point where I gave up".
- "I would like to be able to adapt the pace of assignments".
- "I would like to receive more personalized feedback".
- "I would like to have more educational resources".

The questionnaire was sent to all the repeaters enrolled in the "Fundamentals of Programming" course during two consecutive semesters. The questionnaire was sent at the beginning of the semester by email and it was filled in anonymously.

V. RESULTS

This section presents the results obtained from the exploratory analysis and the questionnaire sent to repeating students.

A. The Behavior in the First Semester is Relevant

Figure 1 shows the evolution of the students semester by semester. A first visual analysis indicates that the behavior of repeating students depends on the results obtained previously. For example, it can be seen that few students re-enrolled the following semester (69+17+46, 23.1%) and that most of them

TABLE II
QUESTIONNAIRE ABOUT FACTORS THAT INFLUENCE HAVING RETAKEN THE COURSE

Category	Item	Question
Difficulty and interest	Q2.1_1	I was motivated to learn programming
	Q2.1_2	Learning to program seemed difficult to me
Resources	Q2.2_1	The teaching materials were clear and sufficient
	Q2.2_2	Teaching materials were well organized and easily accessible
	Q2.2_3	I was able to install the software easily
Assignments	Q2.3_1	The pace of the assignments was adequate
	Q2.3_2	The statements of the assignments were clear and easy to understand
	Q2.3_3	The teaching materials were sufficient to solve the assignments
	Q2.3_4	I felt able to solve the planned assignments
Communication	Q2.4_1	The teacher responded appropriately and in time to my doubts
	Q2.4_2	The feedback I received from the teacher helped me understand the mistakes I had made
	Q2.4_3	The support I received from the lab instructor helped me solve problems
	Q2.4_4	The classroom's discussion forum allowed me to solve my doubts
Personal context	Q2.5_1	I was able to combine the course well with my personal and professional life
	Q2.5_2	The combination of courses which I enrolled in was not adequate

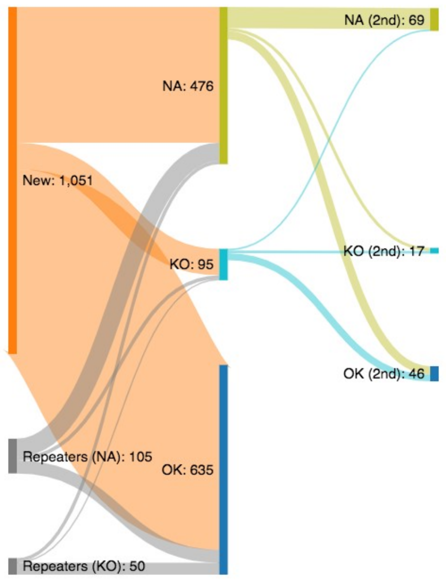

Fig. 1. Results obtained by the students (OK: pass, KO: fail, NA: no attended).

failed again (69+17, 65.2%). These data question whether the current design of the subject is the most suitable for repeating students, since the lack of a specific teaching strategy for them means that few take advantage of their second attempt.

In more detail, Table III shows the breakdown of the students based on the results they obtain in each of the consecutive attempts. From the semester 2017/2 to 2019/1 both included, 1,051 students enrolled in the subject for the first time (labelled as group G1), while 155 took it after not having passed the course previously, of which 50 failed (group G2) and 105 did not attend (group G3). In order to analyze their different paths, we distinguish three subgroups for each of the three groups: those who pass (OK), those who fail (KO) and those who do not attend (NA), labeled as GX-1, GX-2 and GX-3 respectively, where X is their group of origin.

A first analysis shows that the students who had attended but failed (G2) obtained better results than those who did not attend (G3) in their previous attempt. More students tried it one more time (80.0% vs. 47.6%), and even got better results (72.0% vs. 37.1% passed). However, some of these students may come from semesters where the design of the course was slightly different [31], so we will focus on those who start for the first time with the current configuration of the course.

Thus, taking as a reference only the results of the new students (G1), those who had failed the course (G1-2) enrolled more and also obtained better results, but it seems that they did not retake if they did not pass the course on their second attempt. Regarding re-enrollment, from the 411 students who did not attend on their first attempt (G1-3), only 79 (19.2%) retook the course a second time in the analyzed period. In contrast, from the 80 students who failed the first time (G1-2), 29 (36.4%) retook it, a low percentage but considerably higher than the previous one.

TABLE III
STUDENTS ACCORDING TO THEIR ORIGIN AND ACADEMIC RESULTS IN PREVIOUS SEMESTERS

Group	Origin	1st time	2nd time	N (%)	A_4	A_8	Re-enrollment
G1	New			1,051 (87.2 %)	3.10	5.64	108 (10.3 %)
G1-1		OK	-	560 (53.3 %)	3.91	7.70	-
G1-2		KO		80 (7.6 %)	3.56	6.54	29 (36.3 %)
G1-2-1			OK	17 (58.6 %)	3.53	6.94	-
G1-2-2			KO	7 (24.1 %)	3.43	6.29	-
G1-2-3			NA	5 (17.3 %)	1.60	3.00	-
G1-3		NA		411 (39.1 %)	1.91	2.66	79 (19.2 %)
G1-3-1			OK	20 (25.3 %)	3.70	7.15	-
G1-3-2			KO	8 (10.1 %)	3.63	6.75	-
G1-3-3			NA	51 (64.6 %)	1.69	2.29	-
G2	KO			50 (4.1 %)	3.58	6.76	5 (10.0 %)
G2-1		OK	-	36 (72.0 %)	3.89	7.64	-
G2-2		KO		4 (8.0 %)	3.75	7.50	3 (75.0 %)
G2-2-1			OK	2 (66.7 %)	4.0	8.00	-
G2-2-2			KO	1 (33.3 %)	4.0	8.00	-
G2-2-3			NA	0 (0.0 %)	-	-	-
G2-3		NA		10 (20.0 %)	2.40	3.30	2 (20.0 %)
G2-3-1			OK	2 (100.0 %)	3.50	7.50	-
G2-3-2			KO	0 (0.0 %)	-	-	-
G2-3-3			NA	0 (0.0 %)	-	-	-
G3	NA			105 (37.1 %)	2.58	4.36	19 (18.1 %)
G3-1		OK	-	39 (37.1 %)	3.72	7.28	-
G3-2		KO		11 (10.5 %)	3.27	5.45	2 (18.2 %)
G3-2-1			OK	2 (100.0 %)	4.00	7.50	-
G3-2-2			KO	0 (0.0 %)	-	-	-
G3-2-3			NA	0 (0.0 %)	-	-	-
G3-3		NA		55 (52.4 %)	1.64	2.07	17 (30.9 %)
G3-3-1			OK	3 (17.6 %)	3.00	7.00	-
G3-3-2			KO	1 (5.9 %)	4.00	7.00	-
G3-3-3			NA	13 (76.5 %)	1.62	1.85	-

TABLE IV
CHARACTERISTICS OF THE STUDENTS ACCORDING TO THEIR ACADEMIC RESULTS

Group	Gender	Age	Computer Science	Courses	All	Index
G1	M: 838 (79.7 %) F: 213 (20.3 %)	E1: 38 (3.6 %) E2: 474 (45.1 %) E3: 303 (28.8 %) E4: 236 (22.5 %)	NO: 179 (17.0 %) YES: 872 (83.0 %)	ENROLLED: 1.98 PASSED: 1.61	50.1 %	3
G1-1	M: 454 (81.1 %) F: 106 (19.9 %)	E1: 19 (3.4 %) E2: 228 (40.7 %) E3: 172 (30.7 %) E4: 141 (25.2 %)	NO: 95 (17.0 %) YES: 465 (83.0 %)	ENROLLED: 2.00 PASSED: 1.66	81.8 %	6
G1-2	M: 57 (71.3 %) F: 23 (28.7 %)	E1: 8 (10.0 %) E2: 43 (53.8 %) E3: 17 (23.2 %) E4: 12 (15.0 %)	NO: 18 (22.5 %) YES: 62 (77.5 %)	ENROLLED: 2.30 PASSED: 0.99	51.3 %	5
G1-3	M: 327 (79.6 %) F: 84 (20.4 %)	E1: 11 (2.7 %) E2: 203 (49.4 %) E3: 114 (27.7 %) E4: 83 (20.2 %)	NO: 66 (16.1 %) YES: 345 (83.9 %)	ENROLLED: 1.90 PASSED: 0.31	6.7 %	2

Another datum in Table III is the number of assignments submitted (A4, for the first four assignments, and A8 for the first eight, including the previous four). Students who attended and failed (G1-2) submitted almost the same assignments as those who passed (G1-1), 3.56 vs. 3.91 and 6.54 vs. 7.70, respectively, that is, they only slightly decreased their pace in the final four activities, that is, A8−A4 (2.98 vs 3.79). In contrast, those who did not attend on their first attempt (G1-3) submitted very few of the first four activities (1.91), and almost none of the next four (0.75), after teacher's feedback. This fact was also true for students who had already come from a previous attempt (G2 and G3), although in the case of students who did not attend, the number of activities submitted was even lower.

B. *There is not a Single Profile of Repeating Student*

As for the student's features that can explain the differences found in the results of the repeaters (RQ2), it is interesting to observe in Table IV the indicators chosen for the most relevant groups in Table III. For each group or subgroup, it is shown the percentage of men and women, the breakdown by age group, the percentage of students taking the Computer Science degree, the number of courses simultaneously enrolled, and the number of courses passed, in both cases without including the course in question. It also shows the percentage of students who carried out all the assignments and the index (median) of the first assignment that they did not submit, a possible indicator of disconnection from the proposed activity schedule.

It can be seen that in comparison to the group of students who enrolled in the course for the first time (G1), the group of learners who failed (G1-2) had a slightly higher proportion of women and younger students, as well as learners who were not studying Computer Science. They were also enrolled in a few more courses. However, the differences found are not really significant, although they should not be completely ruled out.

On the other hand, the learners who passed the course (G1-1) had much better performance in the other courses enrolled in the same semester, and the majority passed them all (83.0% of courses passed), unlike of those who attended and failed (43.0%) and those who did not attend (16.3%), especially in the latter case. This may indicate that the students who dropped out the course could be actually dropping out all the courses for which they had enrolled, that is, they gave up their studies, what explains why it is not possible to recover them and the re-enrollment index in a subsequent semester is so low.

In addition, it can also be seen that the students who obtained a «Not attended» (G1-3) are the ones who stopped submitting some of the proposed assignments earlier and submitted almost none. The percentage of students who did not attend but performed all the proposed assignments was also considerably lower than those who failed and those who passed (6.7% vs. 51.3% and 81.8% % respectively). Specifically, this dropout occurred in the second assignment, while those who attended and failed or passed did not do so until the fifth or sixth assignment, respectively, i.e. later in the course, after the feedback received from their teacher. Again, it seems feasible to think that part of the students who dropped out of the course did so because they gave up their studies at the beginning of the semester, almost without waiting for feedback from the teacher of the first four assignments.

C. *Questionnaire Results*

Table V shows the results of the questionnaire that the repeating students filled in about their experience in the previous semester, in which they did not pass the course. It is important to note that participation in the questionnaire was very low (N = 38, 20.1%), but this was already expected considering the profile of repeating students, who usually excuse their participation in any non-assessable activity due to lack of time.

As it can be seen, it is worth to remark the high motivation to learn programming of the majority of repeating students who answered in (Q2.1_1, μ = 4.37), while at the other extreme there is the ability to combine the rhythm of activities of the semester with the student's personal and professional context (Q2.5_1, μ = 1.87). The rest of the indicators are quite centered, although some have a greater dispersion, such as the assessment of the feedback received (Q2.4_2, σ = 1.42), indicating a possible bimodality in the responses.

TABLE V
RESULTS OF THE QUESTIONNAIRE. QUESTIONS ASKED IN NEGATIVE ARE INDICATED WITH (NEG.)

Concept	Item	Med.	μ	σ
Motivation	Q2.1_1	5	4.37	0.82
Difficulty (neg.)	Q2.1_2	3	2.97	1.20
Resources	Q2.2_1	3	3.00	1.27
Organization	Q2.2_2	4	3.58	1.15
Tools	Q2.2_3	4	3.76	1.38
Pace	Q2.3_1	3	2.63	1.00
Assignments	Q2.3_2	4	3.24	0.97
Support	Q2.3_3	3	2.84	1.03
Self-efficacy	Q2.3_4	3	2.68	1.16
Teacher	Q2.4_1	4	3.58	1.29
Feedback	Q2.4_2	3	3.24	1.42
Laboratory	Q2.4_3	3	3.53	1.18
Communication	Q2.4_4	3,5	3.53	1.06
Personal context	Q2.5_1	1,5	1.87	1.12
Enrollment (neg.)	Q2.5_2	3	2.79	1.28

A correlation analysis between items in the questionnaire shows that only some are significant ($p < 0.01$). There is a positive correlation between learning resources and the support that these resources offer to carry out the assignments (Q2.2_1 and Q2.3_3, 0.52), between the clarity of the assignments and the support/feedback received from the teacher (Q2.3_2 and Q2.4_1 / Q2.4_2, 0.49 / 0.3 respectively), as well as the support received from the laboratory and the work of the teacher (Q2.4_1 / Q2.4_2 and Q2 .4_3, 0.74 / 0.49 respectively). Regarding negative correlations, they only appear between the perception of difficulty in learning to program and the usefulness of learning materials and self-efficacy (Q2.1_2 and Q2.3_3 / Q2.3_4, -0.33 / -0.45 respectively).

In relation to what the repeating students would like in order to take the course again, clearly the most valued item is to be able to adapt the pace of assignments (chosen by 50.0% of the repeating students), followed by being able to continue from the point where the course was abandoned (26.3%) and having more support from the teacher (15.8%), while having more learning resources is in last position (7.9 %).

VI. DISCUSSION

Before proceeding with the discussion of the results obtained, it is necessary to establish the limitations of the study. We gathered data from the institutional LRS and student responses to the questionnaire. Given that the number of answers in the questionnaire was very low in absolute numbers, the results can only be used as a guide, without the possibility of making generalizations or carrying out deeper analyses.

The results obtained allow us to answer the first research question (RQ1) about the differences that exist between students who repeat the subject depending on their performance in the previous run. Students who do not pass the subject can be classified into two groups, those who try to pass the course until the end of the semester and fail, and those who do not attend and drop out the course in the first assignments. This is consistent with the results described by Porter and Zingaro [41]. The data show that the repeaters who failed the course have a second chance which they can take advantage of, and in fact a high percentage do so. It may happen that they need more time (more than a semester) to pass the course and that they may have had problems at the end, not at the beginning of the semester. On the other hand, the students who did not attend in the previous semester faced the same problems and dropped out again.

Regarding the second question (RQ2), the two abovementioned groups are made up of similar students in terms of gender, age, etc. This also coincides with the results described by [7] on the socio-demographic profile of CS1 students. However, there is a subgroup of students who come from other grades, younger, and with a higher proportion of women, who must take the course as an elective and who fail more, although this also implies that they try more. The fact of offering the same course to all students, regardless of their profile, may be the cause of some of them not being able to adapt to the pace and type of assignments proposed [42]. This circumstance should be kept in mind when designing an intervention with repeating students. Therefore, this design should be aimed at discriminating between students who drop out and those who attend and fail, with the aim of fixing factors that are probably different.

Regarding the perception of the students (RQ3), the results of the questionnaire, although limited by the small number of responses obtained, also suggest some interesting aspects to take into account. It is observed, for example, that the students were highly motivated to learn programming, but that it was very difficult for them to combine the course with their personal and professional life. However, they rather disagree that their enrollment was not adequate, which may indicate that the course seemed to them not to require so much dedication *a priori*, and once they were studying it, most of them recognize that following the proposed pace of assignments was complicated. This is reinforced by the fact that being able to adjust the pace of assignments would be their first option if they could change something in the design of the course. In fact, the weekly pace of assessable assignments, similarly to that described by [2], polarizes students into two groups, those who drop out of the course –who usually drop out in the second assignment–, and those who try, but fail the course –who usually submit until the fifth assignment. Those who pass, if they fail any assignment, it is usually in the sixth of the eight assignments. This polarization effect is also described in mastery learning experiences similarly based on continuous assessment activities [10]. In addition, students who consider that learning to program is difficult, are also less able to do the assignments and are the ones who consider that they do not have enough learning resources to solve them. However, they do not want more resources, but perhaps they need other kind of resources, more visual or practical, such as screencasts, that show them step by step how to solve a problem.

Finally, the early detection of students with problems can be a general strategy, so that it is not needed to wait for such students to become repeaters or they drop out the course early and, in a second attempt, they are better prepared to pass it.

Except for the indicators related to the personal context, the rest are, in general, positive. Thus, the factors of design and teaching strategy of the course are well managed.

VII. Conclusions

In this paper we have presented an analysis of the results of the repeating students of a «Fundamentals of Programming» course, as well as their experience in such course. The repeaters showed two clearly different behaviors: (1) those who in their first semester tried almost to the end and failed, and (2) those who dropped out very early. The performance of these two groups on their second attempt was clearly different, with those who tried and failed the first time passing much more. In any case, offering the same course design to students who repeat it does not guarantee that they will improve their results and not repeat the same mistakes. Therefore, it is necessary to consider some type of different intervention for these students based on their previous experience, profile and needs.

According to the scientific literature on the behavior of repeating students and the problems inherent in teaching programming in introductory courses, some of the most relevant factors that could explain these differences have been identified. A survey was designed and sent, the results of which shed light on the perspective of the students in relation to their experience on the course. The results of the survey show that the students were motivated, but not all of them were able to keep up with the pace of the course or perform the proposed assignments. This was mainly due to circumstances specific to their personal and professional context.

Based on the results of this exploratory research, future work involves the design, implementation and evaluation of an intervention that improves the results of repeaters on their second attempt. This intervention must address the problems detected, especially those related to the difficulty in following the weekly pace of the assignments and taking advantage of what they already did in their first semester.

Acknowledgment

The authors thank the best paper award given by the Scientic Committee of XXVII Jornadas sobre la Enseñanza Universitaria de la Informática (JENUI) Conference as well as its recommendation for the IEEE-RITA journal.

References

[1] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas, "A multi-national study of reading and tracing skills in novice programmers," *SIGCSE Bull.*, vol. 36, no. 4, pp. 119–150, 2004. [Online]. Available: http://doi.acm.org/10.1145/1041624.1041673.

[2] A. V. Robins, *Novice Programmers and Introductory Programming*, ser. Cambridge Handbooks in Psychology. Cambridge University Press, 2019, p. 327–376.

[3] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 Conference on Innovation in Computer Science Education*, ser. ITiCSE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 39–44. [Online]. Available: https://doi.org/10.1145/2591708.2591749.

[4] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming: 12 years later," *ACM Inroads*, vol. 10, no. 2, pp. 30–36, Apr. 2019. [Online]. Available: http://doi.acm.org/10.1145/3324888.

[5] M. J. Marco-Galindo, J. Minguillón, D. García-Solórzano, and T. SanchoVinuesa, "¿Quién tropieza dos veces con la misma piedra en una asignatura inicial de programación?" in *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática (JENUI)*, vol. 6, 2021, pp. 59–66.

[6] Y. Lee and J. Choi, "A review of online course dropout research: Implications for practice and future research," *Educational Technology Research and Development*, vol. 59, no. 5, pp. 593–618, 2011.

[7] A. Robins, "Learning edge momentum: a new account of outcomes in CS1," *Computer Science Education*, vol. 20, no. 1, pp. 37–71, 2010. [Online]. Available: https://doi.org/10.1080/08993401003612167.

[8] A. Luxton-Reilly, *Learning to Program is Easy*. New York, NY, USA: Association for Computing Machinery, 2016, p. 284–289. [Online]. Available: https://doi.org/10.1145/2899415.2899432.

[9] A. Petersen, M. Craig, and D. Zingaro, "Reviewing cs1 exam question content," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 631–636. [Online]. Available: https://doi.org/10.1145/1953163.1953340.

[10] C. Ott, B. McCane, and N. Meek, "Mastery learning in cs1-an invitation to procrastinate?: Reflecting on six years of mastery learning," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 2021, pp. 18–24.

[11] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002, pp. 53–58.

[12] B. N. Miller and D. L. Ranum, "Teaching an introductory computer science sequence with python," in *Proc. 38th Midwest Instructional and Computing Symposium, Eau Claire, Wisconsin, USA*, 2005.

[13] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich, "Revisiting why students drop CS1," in *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 71–80. [Online]. Available: https://doi.org/10.1145/2999541.2999552.

[14] D. Teague and R. Lister, "Longitudinal think aloud study of a novice programmer," in *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148*, ser. ACE '14. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2014, pp. 41–50. [Online]. Available: http://dl.acm.org/citation.cfm?id=2667490.2667495.

[15] K. J. Whittington, D. P. Bills, and L. W. Hill, "Implementation of alternative pacing in an introductory programming sequence," in *Proceedings of the 4th Conference on Information Technology Curriculum*, ser. CITC4 '03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 47–53. [Online]. Available: http://doi.acm.org/10.1145/947121.947132.

[16] H. Lowe and A. Cook, "Mind the gap: Are students prepared for higher education?" *Journal of Further and Higher Education*, vol. 27, no. 1, pp. 53–76, 2003.

[17] S. N. Liao, S. Valstar, K. Thai, C. Alvarado, D. Zingaro, W. G. Griswold, and L. Porter, *Behaviors of Higher and Lower Performing Students in CS1*. New York, NY, USA: Association for Computing Machinery, 2019, p. 196–202. [Online]. Available: https://doi.org/10.1145/3304221.3319740.

[18] N. Hawi, "Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course," *Computers & Education*, vol. 54, no. 4, pp. 1127–1136, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131509003108.

[19] P. Kinnunen and L. Malmi, "Why students drop out CS1 course?" in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 97–108.

[20] U. Nikula, O. Gotel, and J. Kasurinen, "A motivation guided holistic rehabilitation of the first programming course," *ACM Trans. Comput. Educ.*, vol. 11, no. 4, nov 2011. [Online]. Available: https://doi.org/10.1145/2048931.2048935.

[21] F. B. Tek, K. S. Benli, and E. Deveci, “Implicit theories and self-efficacy in an introductory programming course,” *IEEE Transactions on Education*, vol. 61, no. 3, pp. 218–225, 2018.

[22] J. Gorson and E. O’Rourke, “Why do cs1 students think they’re bad at programming? Investigating self-efficacy and self-assessments at three universities,” in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ser. ICER ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 170–181. [Online]. Available: https://doi.org/10.1145/3372782.3406273.

[23] P. Kinnunen and B. Simon, “My program is ok – am I? Computing freshmen’s experiences of doing programming assignments,” *Computer Science Education*, vol. 22, no. 1, pp. 1–28, 2012. [Online]. Available: https://doi.org/10.1080/08993408.2012.655091.

[24] A. Forte and M. Guzdial, “Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses,” *IEEE Transactions on Education*, vol. 48, no. 2, p. 248–253, May 2005.

[25] M. Guzdial and A. E. Tew, “Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education,” in *Proceedings of the Second International Workshop on Computing Education Research*, ser. ICER’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 51–58. [Online]. Available: https://doi-org.bibliotecauoc.idm.oclc.org/10.1145/1151588.1151597.

[26] J. Sheard and D. Hagan, “Our failing students: A study of a repeat group,” in *Proceedings of the 6th Annual Conference on the Teaching of Computing: Changing the Delivery of Computer Science Education*, ser. ITiCSE’98. New York, NY, USA: ACM, 1998, p. 223–227. [Online]. Available: https://doi-org.bibliotecauoc.idm.oclc.org/10.1145/282991.283550.

[27] A. J. Gomes, A. N. Santos, and A. J. Mendes, “A study on students’ behaviours and attitudes towards learning to program,” in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE ’12. New York, NY, USA: ACM, 2012, p. 132–137. [Online]. Available: https://doi.org/10.1145/2325296.2325331.

[28] N. G. Fonseca, L. Macedo, and A. J. Mendes, “CodeInsights: Monitoring programming students’ progress,” in *Proceedings of the 17th International Conference on Computer Systems and Technologies*, ser. CompSysTech ’16. New York, NY, USA: ACM, 2016, p. 375–382. [Online]. Available: https://doi-org.bibliotecauoc.idm.oclc.org/10.1145/2983468.2983492.

[29] K. Wood, D. Parsons, J. Gasson, and P. Haden, “It’s never too early: Pair programming in CS1,” in *Proceedings of the 15th Australasian Computing Education Conference*, ser. ACE’13, vol. 136. AUS: Australian Computer Society, Inc., 2013, p. 13–21.

[30] M.-J. Marco-Galindo, J. Minguillón, and T. Sancho-Vinuesa, “Análisis de la progresión de los estudiantes en una asignatura introductoria a la programación mediante redes bayesianas,” in *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática (JENUI)*, vol. 5. Asociación de Enseñantes Universitarios de la Informática (AENUI), 2020, pp. 69–76.

[31] M.-J. Marco-Galindo and J. Minguillón, “La evaluación formativa como factor decisivo en el aprendizaje online. intervención en una asignatura inicial de programación,” in *Actas del VI Congreso Internacional sobre Aprendizaje, Innovación y Cooperación (CINAIC)*, 2021, p. 677–681.

[32] J. Minguillón, J. Conesa, M. E. Rodríguez, and F. Santanach, “Learning analytics in practice: Providing e-learning researchers and practitioners with activity data,” in *Frontiers of Cyberlearning*. Springer, 2018, pp. 145–167.

[33] G. Kanaparan, R. Cullen, D. Mason et al., “Effect of self-efficacy and emotional engagement on introductory programming students,” *Australasian Journal of Information Systems*, vol. 23, 2019.

[34] Y. Qian and J. Lehman, “Students’ misconceptions and other difficulties in introductory programming: A literature review,” *ACM Transactions on Computing Education*, vol. 18, no. 1, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3077618.

[35] J. Sorva, *Visual program simulation in introductory programming education*. Aalto University, 2012.

[36] A. Vihavainen, J. Airaksinen, and C. Watson, “A systematic review of approaches for teaching introductory programming and their influence on success,” in *Proceedings of the 10th Annual Conference on International Computing Education Research*, ser. ICER ’14. New York, NY, USA: ACM, 2014, p. 19–26. [Online]. Available: https://doi.org/10.1145/2632320.2632349.

[37] S. P. Roche and N. M. Martínez, “Evaluación de entornos de programación para el aprendizaje,” in *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática (JENUI)*, p. 83, 2011.

[38] A. Luxton-Reilly and A. Petersen, “The compound nature of novice programming assessments,” en *Proceedings of the Nineteenth Australasian Computing Education Conference*, 2017, pp. 26–35.

[39] J. Warren, S. Rixner, J. Greiner, and S. Wong, “Facilitating human interaction in an online programming course,” in *Proc. of 45th ACM Technical Symposium on Computer Science Education*, 2014, pp. 665–670.

[40] J. Catterall, J. Davis *et al.*, “Supporting new students from vocational education and training: Finding a reusable solution to address recurring learning difficulties in e-learning,” *Australasian Journal of Educational Technology*, vol. 29, no. 5, 2013.

[41] L. Porter and D. Zingaro, “Importance of early performance in cs1: two conflicting assessment stories,” in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 2014.

[42] J. Q. Dawson, M. Allen, A. Campbell, and A. Valair, “Designing an introductory programming course to improve non-majors’ experiences,” in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 26–31.

María-Jesús Marco-Galindo received the degree in Computer Science from the Universtiat Politécnica de Catalunya (UPC) in 1999. He also holds a PhD in e-learning (2013) and a Master’s degree in Information Society (2003) from the Universitat Oberta de Catalunya (UOC). At the UOC, he has been an associate professor since 1999. Currently, she develops her teaching activity in the areas of programming languages and communication skills. His research in the LAIKA research group (Learning Analytics for Innovation and Knowledge Application in Higher Education) focuses on different topics related to e-learning and e-assessement tools, feedback and portfolios

Julià Minguillón holds a PhD in Computer Engineering (Universitat Autònoma de Barcelona, 2002). In 2001 he joined UOC as a lecturer in the areas of programming languages, computer graphics, statistics and data mining and visualization. He belongs to the LAIKA research group and his research interests include the description and standardization of educational content and the process of learning through ontologies, semantic repositories of learning objects, modeling the behavior of users of a virtual learning environment through web mining techniques, and more recently, educational data mining and learning analytics.

David García-Solórzano received his BS degrees in Multimedia and Computer Science Engineering from the Universitat Ramon Llull (URL) in 2005. At the same university, he earned his MS degrees in Multimedia and Computer Science Engineering in 2007 and 2008, respectively. In 2013, he obtained a PhD degree from the UOC. At the UOC, he has been a lecturer since 2008. His research interests include topics related to learning analytics, information visualization, self-regulation and teaching programming online.

Teresa Sancho-Vinuesa received the degree in mathematics from the Universitat of Barcelona (UB), Spain, in 1990, and the Ph.D. degree in electronic engineering from the URL, Spain, in 1995. She is a full professor at the Universitat Oberta de Catalunya (UOC), where she combines teaching and research in the area of mathematics for engineering degrees. She also leads LAIKA research group, where she is currently concentrating her research efforts in the use of learning analytics for the improvement of online education and learning, and more particularly in the evaluation and feedback processes in mathematics.