An empirical study of students' perceptions on the setup and grading of group programming assignments

Aivaloglou, E.; Meulen, A.N. van der

An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments

EFTHIMIA AIVALOGLOU, Leiden Institute of Advanced Computer Science, The Netherlands and Open Universiteit, The Netherlands
ANNA VAN DER MEULEN, Leiden Institute of Advanced Computer Science, The Netherlands

Courses in computer science curricula often involve group programming assignments. Instructors are required to take several decisions on assignment setup and monitoring, team formation policies, and grading systems. Group programming projects provide unique monitoring opportunities due to the availability of both product and process data, as well as challenges in team composition, with students of varying levels of prior programming experience. To gain insights into the experiences and perceptions of students about the assignment setup and grading policies in group programming projects, we interviewed 20 computer science students from four universities. The thematic analysis highlighted factors in group composition that are considered important, as well as advantages and disadvantages of the self-selection of the teams. It also indicated three grading strategies experienced by the students, namely, being assigned the same group grade, individual grades distributed by the instructor, and grade distribution determined by the team, with perceptions about them varying greatly. Several practices for monitoring team contributions were identified. Checking the source code repositories was considered useful in recognizing slacking members, but automated metrics are not always representative of the work distribution. The analysis also uncovered student perceptions on the grading factors for programming assignments, including coding efficiency and skill.

CCS Concepts: • **Social and professional topics** → Computing education;

Additional Key Words and Phrases: Computing education, group projects, programming, grading

ACM Reference format:
Efthimia Aivaloglou and Anna van der Meulen. 2021. An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments. *ACM Trans. Comput. Educ.* 21, 3, Article 17 (February 2021), 22 pages.
https://doi.org/10.1145/3440994

1 INTRODUCTION

Computer science students typically participate in several group projects during their studies. Several reasons motivate the group assignment setup: forming groups can help courses in scaling to the growing number of computer science enrollments while, at the same time, provide the

Authors' addresses: E. Aivaloglou, Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands, Open Universiteit, Heerlen, The Netherlands; email: e.aivaloglou@liacs.leidenuniv.nl; A. van der Meulen, Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands; email: a.n.van.der.meulen@liacs.leidenuniv.nl.

1946-6226/2021/02-ART17 $15.00
https://doi.org/10.1145/3440994

opportunity to students to work on software projects that are of larger scale than individual course projects can be. Further, group work can be used as an instructional strategy, included in education because of its benefits for the learning process of domain-specific knowledge [24, 25, 40]. Participating in group assignments also enables students to practice their collaboration skills [36], which are recognized as important not only by computer science faculty [45], but also by the software development industry [19, 30, 38], with current industrial trends promoting cooperative working techniques such as peer feedback [39], collaborative software design [2], shared code ownership, and pair programming [44]. The "ability to work with others and check your ego at the door" has been identified as one of the most important attributes desired of software engineers [19], as even early career software developers "need to know how to work with other people" [12]. At the same time, it has been found that communication and teamwork skills are areas where graduating computer science students frequently fall short of the expectations and work requirements of industry [12, 37].

The setup of group assignments greatly affects their success in terms of group performance, learning, collaboration patterns, and member accountability [20], while team formation policies strongly affect individual contributions and team collaboration [13, 33]. Students' experience of group work is impacted by grading strategies [11], where work and grade equity concerns negatively impact students' attitudes toward working in a group [8].

The topics of group work setup and grading have been researched in the education domain. In computing education, however, group programming projects have some unique characteristics. First, students enter computer science courses with varying levels of prior programming experience, which has been found to strongly affect their learning performance [1, 47] and could impact team formation and work distribution. Second, in contrast to early recommendations against establishing groups with less than three members [13], computer science students could be given the opportunity to apply practices like pair programming in their assignments. Finally, while in other types of course projects the only data available to the instructors are the project deliverables, software projects can also make process data visible [15], which can be used to monitor student contributions [6, 32]. Automated tools have lately been proposed for providing feedback to instructors [22, 42], but have not been evaluated from the students' point of view.

The goal of this article is to explore how these characteristics of programming assignments affect group work and its monitoring and grading through the perceptions of computer science students. Specifically, we are examining the following research questions:

RQ1 What are the experiences and perceptions of students on the setup and team formation policies of programming assignments?

RQ2 What are the perceptions of students on the grading strategies applied to group programming assignments?

RQ3 Which techniques are employed for monitoring team contributions in programming assignments, and what are the perceptions of students about them?

To answer our research questions, we interviewed 20 final-year bachelor's and master's students from four universities about their experiences on group programming assignments of varying group sizes and scopes, in which they have participated throughout their studies, and about their perceptions on assignment setup, team formation policies, and grading.

Our thematic analysis revealed that large differences between team members' prior programming experience, skills, and commitment are considered problematic. The self-selection of the team composition was found to have advantages and disadvantages, such as missing the education value of new groups. Several practices for monitoring team contributions were discussed. The source code repository was thought to be a good indication of member contributions, but challenging in

finding representative metrics, and should be combined with other monitoring practices. Students also thought that peer reporting could be useful, but is susceptible to colluding team members, lacks transparency, and is ineffective in informing on slacking members because of social consequences and lack of incentive. Three grading strategies were described: being assigned the same grade, individual grades distributed by the instructor, and grade distribution determined by the team, with perceptions about them varying greatly. Students also expressed perceptions on the meaning of the grade and the grading factors for programming assignments, including coding efficiency and skill, code functionality, and quality.

2 BACKGROUND AND RELATED WORK

2.1 Group Work at the University Level

Group work at university-level education is a well-researched area. First, a wide range of studies attempt to gain insight into the overall value of group assignments, including their positive and negative aspects and the influencing factors [3, 13, 27, 28, 46]. The emphasis of these works often lies on the importance of group assignments in preparing students for the labor market. Some of the earliest insights into this research line include recommendations on team formation practices and grading policies, in the form of avoiding peer evaluations for grading, discouraging students from forming their own teams, and ensuring heterogeneity between the team members [13]. The formation and setup of groups, the related issue of differences in motivations and contributions between team members, as well as grading are central themes [3, 27, 28, 46].

Second, a different set of research lines focuses on group assignments as an instructional strategy. This includes team-based learning [33] and collaborative learning [25, 40]. Collaborative learning is presented as an educational approach with confirmed benefits for the learning process itself. Actively working together towards a mutual learning goal [43] has been found to be more effective, compared to individual approaches, in certain types of learning [24, 25]. This can be understood from the cognitive load theory, which describes the limitations of the individual working memory during complex tasks and the benefits of sharing this task in a group [24]. Although studies in this line of work are more primarily aimed at understanding the effects of student collaborations on performance in terms of domain-specific knowledge [36], attention is also paid to circumstances that foster effective collaboration, which include complexity of the task and asymmetry in knowledge or expertise between members [25]. The team-based learning paradigm similarly stipulates that the instructor oversees the formation of groups to ensure that the groups have adequate resources and that member coalitions are avoided [33]. Team formation was found to be critical but problematic in a recent application of team-based learning in a computer science course. Students agreed that they did not want to select their own teams, but disagreed on how team formation should be done [23]. Benefits of collaboration are found when facilitating measures are taken, such as scripted learning environments including rules for communication and coordination [25].

The current study fits into the first line, looking at students' group work as an occurring educational activity within computer science and not as an instructional strategy; however, both research directions form the basis of understanding students' collaboration.

2.2 Grading of Group Work

Grading, or evaluation and assessment, is one of the most important themes within research on group assignments [21], especially as a concern of the students [27, 28]. The issue with grading in group assignments can be understood from the convention in higher education to base rewards on the project outcome and not on the process leading up to this outcome. This might be appropriate for individual projects, where both the input and the output are determined by the individual, but not in group projects, where there is a less clear link between individual inputs and the output

[21]. The common concern in grading group work is having all group members receiving the same grade, while they contribute unequally [27, 28]. Students indicate several drawbacks of the same grading policy, including slacking members, motivated members having to compensate, and ratting out being avoided [11]. Interestingly, perceptions of students and instructors have been found to differ; while students perceive individual contributions not to be taken into account, faculty members of the same institution were of the impression that they were [27, 28].

Another solution is to allocate individual grades within a group assignment based on individual contributions. There is debate on the best approach to determine these [21], with a wide range of options being available [14]. Contributions can be tracked by instructors using various systems or tools. Since this is often considered difficult, depending on the discipline, peer- and self-evaluations by the students are a common approach with several recognized issues: Students tend to base their ratings on perceived ability instead of real contribution and are influenced by gender bias [29]; they are self-biased, rating their own contribution higher than that of other group members [21]; and might distribute the ratings equally or be too hard on themselves, suggesting that their perceptions of effort are not objective [18].

2.3 Group Work & Grading in Computer Science

Research on group assignments in computer science confirms known issues, including difficulties with peer assessments [17], and suggests particular themes related to specific features of computer science assignments such as the visibility of process data [6, 15, 32].

Hayes et al. discussed the use of common grading techniques in group software engineering projects, including assigning the same grade, self reporting, peer reporting, pop quizzes, and cross-validating with the results of individual work [17]. From the instructors' perspective, they agreed with earlier findings from outside the software engineering field that teams would often collude, and peer assessment was ineffective because students are reluctant to inform on their teammates. Clark et al. also reported on problems with self and peer assessment in a software engineering project, including reluctance to report slacking members, attempts to manipulate reported numbers, and applying pressure on team members [10]. A recent study on evaluation methods of student software development group projects revealed that students' preferences for evaluation methods are influenced by their study year, familiarity with teammates, and happiness with the team experience [41].

Software projects can be monitored through the utilized collaboration platforms, with services such as GitHub making student activities transparent and enabling continuous monitoring of project progress and student participation [48]. In the industry and the open source software development world, various metrics have already been proposed for quantifying developer contributions using the source code repositories [15, 31]. Measuring productivity in software development teams can have unintended consequences such as warping developer's incentives, making them change their behavior, and self-monitoring their every action [26].

In the computer science education field, Patton and McGill suggested the use of student portfolios and software quality metrics expanded with pedagogical metrics, such as language vocabulary [35]. Information from the source code repository and static code analysis metrics have been used to assist students in learning to program [7], to monitor member participation and contributions in teams [9, 34], as well as to monitor and improve quality of programs [32]. In addition to the source code repository, information from the teams' Kanban project management boards have been used and compared to peer evaluations. It was found that students' commits were associated with peer evaluations of their contributions [6]. Automated tools have also been proposed for monitoring student collaboration and contributions, including tools utilizing students' wikis and software version control system (SVN) repositories [22], and analyzing online team discussion transcripts to visualize team mood, role distribution, and emotional climate [42].

3 METHODOLOGY

The goal of this study is to explore experiences of computer science students with group programming assignments and their perceptions on assignment setup and grading. To this end, 20 semi-structured interviews were conducted with students from four universities. The following paragraphs describe the interview and data analysis process.

3.1 Interview Process

For our study, we needed to ensure to include participants that have a range of experiences with group programming assignments to describe to us and to inform their perceptions. For this reason, we focused on recruiting students who are towards the final stages of their study years, i.e., final-year bachelor's students and master's students of computer science departments. Participants were recruited from four public research-intensive universities in the Netherlands, including a technical university and a university that offers distance learning opportunities. Participants were invited during their classes and lab sessions, as well as through Slack and other student communication channels.

We conducted 20 interviews, with at least four students from each university. Twelve students were final-year bachelor's students, while eight students were master's students. Four of the master's students had completed their bachelor's degrees in other universities, three of which in other countries. Our participants self-identified as originating from (alphabetically): Bangladesh, India, the Netherlands, Switzerland, Ukraine, and the U.S. Five participants self-identified as being female and 15 as being male. We interviewed participants who declared being autistic, as well as students with speech disorders. The youngest participant whose age is known was 20, and the oldest was 33 years old. The self-reported expected grade of the participants for the bachelor's or master's degree that they were pursuing varied from 6.5 to 9, with a mean value of 7.5 out of 10, with 5.5 being the passing grade. After 20 interviews, no new information appeared to emerge, reaching saturation [16].

We used a semi-structured interview protocol consisting of questions in five topics: (1) student background, (2) experiences with group programming assignments, (3) perceptions on assignment setup and group formation policies, (4) experiences with grading and monitoring techniques, and (5) perceptions on grading and monitoring techniques. The questions within the second and third topics correspond to RQ1, while the questions in the fourth and fifth topics correspond to RQ2 and RQ3. To capture a broad view of student experiences for our exploratory study, the questions were formulated to include group assignments of varying group sizes and scopes. The first question of the second topic was *"During your studies, have you participated in programming assignments that were done in groups of two or more people?"*, enabling the students to describe both pair programming assignments and assignments of larger group sizes, as well as both course assignments and longer-term ones.

The first two interviews, of one male and one female master's student, were used to inform the study scope and to refine the questions of the protocol, which remained unchanged for the other interviews. Ten of the interviews were voice-recorded face-to-face interviews, and 10 were conducted through recorded Skype sessions. The average length of the interviews was 23 minutes. The interviews were transcribed using automatic transcription software and were afterwards manually corrected. The transcripts were anonymized and used for subsequent processing.

3.2 Data Processing

A thematic analysis was conducted on the transcripts [5]. The approach was partly theory-driven, starting with themes directly derived from the research interest; and data-driven, generating

appropriate labels within these themes from the data [4, 5]. First, themes were determined and labels were developed. Six themes were determined based on the research questions of the project, closely in line with the topics of the interview protocol: student profile, group assignment setup, experience with grading and monitoring techniques, and perception on grading and monitoring techniques. Next, two researchers labeled independently two interviews, developing labels based on the data within the determined themes. For example, within the theme "experience with grading and monitoring techniques," the label "same grade" was generated. The researchers compared their labels and discussed the themes and assignments of labels. To continue developing appropriate labels, both researchers labeled independently four additional interviews. The researchers again discussed their labels, reaching an agreement on fixed labels within each theme.

Second, these labels were used to code all interviews. The interviews were divided between the researchers. Doubts as to which label was appropriate for a given excerpt were discussed.

Third, the information was integrated across participants for each theme. Where applicable, information from one theme was taken together with information from another theme. For example, in describing the experience with group assignments, the profile of the students (institution and current degree) was included.

4 RESULTS

The participants have been assigned random numbers and are referred to as S1 to S20. The quotes that are included in the description are verbatim. To protect the anonymity of the participants, identifiable information has been suppressed and personal pronouns have been converted to the male form.

4.0.1 Experience with Group Assignments. Almost all students have participated in several group assignments during their studies as part of different courses. Only one student indicates having no experience at all, and one other student being part of only one group assignment (with one other person) so far. Both of these students are undergraduate students from the distance learning university. Two other students from this university are part of the sample: an undergraduate student, who describes multiple group assignments with more than one other member, and a postgraduate student who mentions limited experience. Overall, the inclusion of group assignments in their studies is treated by the students as a given, as indicated by the response *"Yeah, of course. We would have like multiple group projects"* (S6).

Students mention having worked in groups of various sizes, referring both to groups of two people and, commonly, of larger size. Nine participants explicitly refer to groups of two people, three of whom identified it as common case, while 15 participants described having worked in groups of more than two people, specifically in team sizes of three (seven participants), four (three participants), five (six participants), and six or more students (three participants).

The students elaborate on group programming assignments of varying scopes, which spread over all years of their computer science studies. The most commonly mentioned case of group assignments were course assignments, both small-scale ones and semester- or quarter-long ones. Larger scope group projects were mentioned by four students of two universities, who described having worked in groups for their bachelor's thesis projects.

4.1 RQ1: Assignment Setup

The students described their experiences and perceptions about three specific aspects related to assignment setup: the team formation policy (where a distinction was made between self- or instructor-assigned groups), the group composition and its influence to group work, and the degree of instructor involvement in their work strategy. Table 1 presents a summary of the students' perceptions for each of those aspects.

Table 1. RQ1: Summary of Students' Perceptions on Assignment Setup

Assignment Setup	Student Perceptions
Self-Assigned Groups	(+) Benefits of knowing each other's strengths and work process
	(+) Similar motivation, commitment, skill level
	(+) Easier communication and having fun
	(–) Challenges in group selection process
	(–) Collaboration issues because of familiarity
	(–) Risk of ending up with the same problematic group
	(–) Reduced opportunities of learning to collaborate
	(–) High-skilled students sticking together
Instructor-Assigned Groups	(+) Lifts the burden of turning people down or finding teammates
	(+) Educational value of new groups
	(–) Risk of pairing with incompatible students or dropouts
	(–) Risk of different levels of experience or motivation
Group Composition	Neutral/negative impact of differences in study direction
	Negative impact of differences in skill level (both lower and higher)
	Importance of similarity in motivation and commitment
	Effects of familiarity, cultural background, group size, gender
Instructor Involvement	Involvement in organization of the team work undesirable/unnecessary
	Involvement in assignment of sub tasks or monitoring contributions
	Mixed views on instructor prescribing tasks or dividing up the work

4.1.1 Self- or Instructor-Assigned Groups. The majority (15) of the students describe how groups are being assigned, making the distinction between selecting their own group and the instructor assigning the group. Nine of these students refer to both strategies, three refer to only self-assigned groups, and three only to instructor-assigned groups. The 12 students who discuss selecting their own group see both advantages and disadvantages. Advantages lie first in knowing each other. This is beneficial, because it means being familiar with all group members' strengths and with the group dynamic and process of working together. Second, there are advantages in being similar in terms of motivation and commitment, mindset, and skill level. A self-selected group implies easier communication and having fun. As S4 explains, *"I absolutely say go with people you know, and especially your friends if they know what they're doing. Um, because then you already have the social connections and the expectations of each other."* One student described pairing with a specific classmate across courses, which enabled them to split the projects between them.

Students see disadvantages in selecting their own group in, first, the process of selecting the group. They describe that not everyone is good at this, that it can remain random for students who do not have friends, as well as problems at the start of the study, when people have not formed social connections yet. Compatibility can also be difficult to estimate, as well as the choice for similar or dissimilar skills. Second, the collaboration within a self-selected group can be difficult, because the level of comfort can be too high (resulting in a too relaxed approach) and because of the risk to end up in the same group with the same issues several times. The importance of learning to collaborate is also taken into consideration: *"so of course I'd like to select them myself, but I think there's definitely some advantage to the instructor assigning people, because then you need to work with different people and it's actually a pretty important skill"* (S17). Finally, another process that is mentioned by two students is the tendency of high-skilled students to stick together: *"Um, and also, the ones that are really the ones that are really productive, they tend to find each other and they just stick to themselves"* (S7).

Of the 12 students who describe instructor-assigned groups, four students are positive and mention conditions under which such groups work. These include the instructor matching the students on experience or skill level, the instructor being present to address issues, and the aspect of "being lucky": *"and I think I'm very lucky because I don't really think there are people in the computer science Bachelor or Master programs here who aren't fun to work with"* (S3), *"it could be like you could know different pupils and maybe in case you are lucky then you will get to learn something new from them as well"* (S6). Advantages are described, first, for the process of instructor-selected groups: it avoids having to turn people down with whom students prefer not to work, and it is beneficial for those without social connections or friends. Second, concerning the collaboration itself, educational value can be found in new groups, such as having to step up and learn new things, gaining knowledge from each other, and learning to work with different people: *"And that's also one learning curve [...] like it's not possible to always work with the same group of people or with your friend, you know, so you have to adjust, uh, and adapt on how different people work because in the future, well you will be working in like different organizations you have to deal with these things"* (S6). However, disadvantages can occur when paired with people with whom they are too incompatible or who drop out (recognized as a common problem in the first year of the undergraduate studies) and when members have different levels of experience or motivation.

4.1.2 Group Composition. Overall, 11 students mention group composition as a factor in group assignments. Group composition refers primarily to differences between group members that can affect the collaboration. Most often mentioned are differences in study direction. The influence of this is either seen as negative (students from certain directions cannot do certain required tasks) or neutral (group dynamics and roles are determined by study background, but not in a negative way). In terms of the differences between the skill level of group members, two students mention that instructors stress the benefit of different skills but that they themselves think similar skills work equally well or better. Second, although the presence of members with a lower skill level is considered problematic, the same applies to group members with higher skill or commitment level. S5 explains that *"I feel working too hard is almost more problematic than working not hard enough because not hard enough means the rest of the group learns something from it still, maybe more even, or if one person works very hard you take away all of the work from others. And from an educational perspective, I always get very upset with that because the group assignment is made to [teach] people to program in groups. And you're really taking that away if there's one person who, who takes all the work away."*

Differences between members in their motivation and commitment are considered very important, affecting the overall group dynamic and collaboration: *"if you don't want the same thing, then it's really tough. Then there's like not really communication possible. The ones that don't want to work, they don't want to communicate"* (S1). Finally, several other factors are mentioned: the extent to which members know each other, which can both facilitate coordination, stimulating help between members, and be uncomfortable if a member does not contribute, cultural background, affecting the understanding of group agreements, group size, with increased chances of slacking when there are more than two members, and gender, which is considered either not having an effect or an unclear effect.

4.1.3 Instructor Involvement and Prescribed Work Strategy. Seven students who discuss the role of the instructor in group assignments bring up different aspects and perceptions. These include the instructor being concerned with the organization of the team work, which one student considers not preferable and one student considers not needed. Other students do see the potential of an active involvement of the instructor in the form of the assignment of sub tasks, putting a

Table 2. RQ2: Summary of Students' Perceptions on Grading Strategies

Grading Strategy	Student Perceptions
Same Grade for All Team Members	(+) Good enough/fair, if policy known in advance (+) Allows for a team feeling of taking responsibility for the project (+) Promotes improving teamwork skills (−) When slacking members: unfair, demotivating, exploitation possibility (−) Unfair because of the varying quality of student programs (−) Instructor not caring about work distribution
Individual Grades Distributed by Instructor	Can be fair it multiple monitoring practices are combined (−) hard to assess/quantify individual contributions (−) assessment hindered by colluding teams and non-reporting slackers (−) overly complicated grade distribution process (−) one-sided, lacking transparency
Individual Grades Determined by the Team	(+) Positive reinforcement, gives the group some freedom (−) Social consequences (−) Student reluctance to use it, required directness
Perceptions on Grading Factors	Software functionality, quality, and coding style Teamwork and group communication Effort contributed versus efficiency and prior programming experience Having an overview of the project and learning Requirement for evaluation rubrics, transparency and feedback

group member in charge, helping in forming the team, keeping track of individual contributions, or providing a general framework on how to work together. The students also mention the involvement of the instructor with the content of the work and their appreciation of help with the actual programming.

Students also talk about the specific situation of the instructor prescribing the work strategy within a group. Some students experienced, in certain courses, the instructor prescribing what to work on (dividing up parts) or how (using certain collaboration tools). One student in particular is not positive about this: *"all of a sudden you feel like you're doing organizational work and you're doing collaboration because it's forced and it's no longer natural. You know, I much prefer to have a team find their own way of, you know, organizing themselves"* (S3). Two other students do not talk from experience but agree with not seeing any merit in this approach, because they prefer dividing up the tasks themselves and because, otherwise, a view of the complete assignment is missed. Two students do see the potential value; as one of them considers *"I think if a teacher says, 'okay, you do this and you do that,' it takes away a lot of freedom. But it's also, it will improve their learning I think, because if you do something that you're not good at, eventually you will become good at this. So it has its pros and cons, I believe"* (S13).

4.2 RQ2: Grading Strategies

The analysis indicated three grading strategies experienced by the students, namely, being assigned the same group grade, being assigned individual grades that are determined by the instructor, and grade distribution determined by the team. Student perceptions on the meaning of the grade and the grading factors for programming assignments are presented separately in Section 4.2.4. Table 2 presents a summary of the students' perceptions for each of those grading strategies.

4.2.1 Same Grade for All Team Members. The most commonly reported grading strategy, from 15 students from all four universities, was assigning the same grade to all team members. Some students considered it as the default option, when there is not *"much discussion"* (S1) about the grading strategy. Students recognized that, when this strategy was used, there could still be differentiation in the grades in exceptional cases of slacking members, if *"someone files an official complaint"* (S17), or *"if one teammate committed fraud or something"* (S13). Perceptions about the same grading strategy varied greatly, but a universally identified prerequisite was that slacking members should be managed, through practices discussed in Section 4.3, and especially through filing complaints. As S19 explained it, *"it's okay if someone does more than the other, but it has to be within proportions."* Another identified prerequisite was that the strategy should be made known in advance.

Several advantages of the same grading strategy were recognized, including that it allows for a team feeling of taking responsibility of the whole project and that it is a fair and good-enough strategy if it is known in advance. If there are imbalances in the contributions, teams can still self-regulate: *"we are already in our twenties. We must be old enough to say, 'okay, you are doing not enough work, you are going to get kicked out or you will do more work than you did last time"'* (S14). The same grading strategy also promotes improving teamwork skills: *"we also need to learn how to work together"* (S19). Some advantages of the same grading strategy were associated with disadvantages of individual grading. First, it allows focusing more on the project, as explained by S10: *"it feels more like you are all trying to produce something of high quality and not just like you trying to prove that you did enough [. . .] it's not like this survival mode where you need to like make sure do my teammates like me? Like am I, do I look motivated? Did I have like, committed anything today? You know, like it's more keeping up appearances."* Second, it was considered too hard to differentiate between contributions in programming assignments, and getting the same grade was found to be more transparent than justifying individual grades and wondering which criteria were used.

Examining the disadvantages of the same grading strategy, most commonly mentioned was that it can be unfair and demotivating in the presence of slacking members. S6 explains: *"if you're a group of three and two people who worked really hard, one person didn't do anything at all, but at the end if you got like excellent grade. So he also got the same grade without doing anything. And if you got poor marks because of him, two others have to face these things. It's not fair to anyone as a group."* It also forces team members to fill in for the slacking members, as described by S10: *"it also like invites even more that if you do end up with a partner that is not like willing to work, then either you can choose either you fail as well or you do everything and he gets the grade as well."* The same student described an experience where this was exploited: *"at some point he [his teammate] announced to me, he's going on vacation like two weeks before the deadline. It's not finished. So in the end I finished it, I handed it in, and then he got his 90 as well, even though he was already on the beach somewhere."* Receiving the same grade was also considered unfair because of the varying quality of the programs that students produce. For example, *"if you have done part of the assignment and it works perfectly, but the others delivered something that was broken or it didn't work. So the whole program didn't work, but you did your best, so you also get a bad grade based on the work of others"* (S13). Finally, assigning the same grade was interpreted by two students as the instructor not caring about the work distribution.

4.2.2 Individual Grades Distributed by the Instructor. The strategy of assigning individual grades to the team members was discussed by 10 students. Five of these, from four universities, described having experienced it and mentioned combinations of practices (examined individually in Section 4.3) that were applied and affected the grade distribution. The combination, for example, would include examining the produced source code along with the description of contributions in a report and a final presentation or, at a distance learning university, a report on the contributions

along with a group presentation and a private interview. A few students expressed that it could be possible to assign fair individual grades using a variety of information sources, to *"reflect that someone is pulling more weight and someone doesn't"* (S2), while S5 expressed that *"I do feel, especially in early projects that there should be a distinction between team members."* A recognized prerequisite for this strategy is to combine multiple monitoring practices, because neither input from the students nor product metrics can give the full picture; the first practice is susceptible to colluding members and defaming, and the latter requires *"making sure that people can explain all of it or at least more than only what they literally committed"* (S1).

Several negative perceptions were expressed about the strategy of assigning individual grades. First, the most commonly mentioned drawback was that it is hard to quantify the effort of each member and how it is reflected in their contributions. As S10 expressed, *"it's almost impossible to objectively grade even if you're really doing your best."* This was also found to be affected by the varying prior programming experience and skills of the team members, as well as by the different perceptions of the meaning of the grade, discussed in Section 4.2.4. Students revealed that the individual grades might not reflect the contributions because the instructor is not provided with the necessary information. This might be due to colluding teams, as S20 explains: *"I think it's very difficult for a teacher to know how much work the people do, so they decide, we also decide in our group how we're going to present things to the teacher. Um, so a lot of the time teacher just doesn't know what's also true,"* or because of non-reported slackers: *"like how do you grade someone that did way less work, and somebody that did their work instead? Well, you don't, I mean, you wouldn't find out about it anyways"* (S7). Second, a drawback of individual grades is that the grade distribution process can be overly complicated. As S3 explains, *"I hate the instructors who [...] overly control and then as a result make it this weird grade distribution that isn't really equal."* It is also sometimes perceived as one-sided, without feedback substantiating the resulting grade or explaining the applied distribution. Finally, the grade distribution process is sometimes found to lack transparency, since *"practical implications are really hard to justify"* (S2).

4.2.3 Individual Grades Determined by the Team. Three students from two universities described a project where the team members had the opportunity to directly affect the grade distribution according to two strategies. In the first strategy, group members could assign their individual grades that needed to have the group project grade as their average. S20, who reported on this strategy, did not make use of it, saying that *"we decided that everybody would just get the same grade."* According to this student, this strategy can cause fights between students, especially in the case of highly motivated members, and it does not lead to fairer grades, because students are too friendly to use it. In the second strategy, students had the opportunity to assign bonus points to selected team members. S4 described the process as *"at the end of the final presentation, the instructor would allow us to select one member who did exceptional work and give them a half a point bonus."* In his case, this project took place in the first year of the computer science studies, *"it's a very tumultuous year with great differences between the skills of people, people leaving. Um, so I think giving some people a bonus and requesting a certain report on how preparation went was a very good way of solving this."* S5 added that he liked that this strategy gave the group *"a bit of freedom"* and described it as positive reinforcement.

Negative aspects of this strategy are mainly related to social consequences and the directness that it requires from the group members. S5 mentioned that *"someone still holds a grudge against me for saying, 'yeah, you don't deserve that point because you did not do enough work'"* and stressed the social consequences that this can have, since it *"can really hurt the relationships between people within the same year."* Moreover, deciding on bonus points *"becomes problematic if there's a group which really doesn't work together"* (S4).

4.2.4 Perceptions on Grading Factors. A topic that often came up during the interviews was the meaning of the grade in programming assignments and the factors that the grade reflects. Half of the students described their perceptions on what the factors should be. The most commonly mentioned grading factor was that the delivered software offers the required functionality and, if needed, is implemented in the way specified by the assignment. This factor was considered so important for some students that they expressed that *"I prefer it if the programming assignments were just pass/fail"* (S7) and *"if you see from the code that it's sufficient that they've done sufficient work, I don't even think you need to go further"* (S3). Other grading factors that are being applied relate to software quality and coding style, which a student described as a bad factor because of its subjectivity and inconsistencies between the graders. Three students discussed teamwork and group communication as a grading factor that is used *"because the idea is that you work together in a real world so you should [...] start learning"* (S9). A counter-argument mentioned for this was that in the industry there is a supervisor to distribute the workload and keep members accountable, which is not the case in student teams.

Conflicting opinions were expressed about what the grade should be influenced by. The effort that each team member puts in the project was important for S18, while others expressed that the focus should be on the end-product and the complexity of the work that they committed and *"should never be impacted too much by an assessment of how much someone has worked"* (S3). This relates to the factor of students' efficiency in programming, especially when attributed to prior experience. While it was found to be natural that the grade represents this, S10 questioned the higher grade that an experienced teammate got: *"what was it based on? Because I really spent a lot of time on it as well, but I'm just less quick than he is."* Further, two students expressed that the grade should acknowledge having an overview of the whole project and understanding and learning from teammates' code: *"the teacher could require team members to grasp 100% of the code, whether they wrote it themselves or not"* (S16). Finally, the availability of evaluation rubrics, transparency, and feedback were expressed as requirements for the assigned grades by three students. S10 advised teachers to *"be really transparent about how you decide what, what they're graded for, even at all. Like what are the factors? Is the teamwork, is it the quality of your code? Like what are you looking at? Uh, and then in the end, explain their grade."*

4.3 RQ3: Monitoring Team Contributions

Several practices for monitoring team contributions were discussed during the interviews: checking the collaboration platform, peer reporting, presentations and verbal assessments, individual interviews, and reporting on slacking team members. Table 3 presents a summary of the students' perceptions for each of those monitoring practices.

4.3.1 Checking the Collaboration Platform. The majority (17) of the students describe the practice of team contributions being monitored through information from the source code repository. Seven students (from three universities) experienced this practice. The information utilized was mostly related to the git history, along with metrics related to commits and lines of code, as well as comments, issues, and the source code itself. Most students described that the repositories were checked at the end of the projects to ensure that the team contributions were balanced. Imbalances usually led to a discussion initiated by the instructor, with the possible result of failing the course. An exception is the experience of a student who was failed without any communication by the instructor, which he described as the *"worst experience in my entire computer science [studies]"* (S1). In some projects, the repository was not only checked at the end, but was regularly monitored by the teaching assistants, who *"occasionally checked out our standing merge requests, saw the reviews and the discussions on them"* (S4). Information from the repository was also sometimes used to assign

Table 3. RQ3: Summary of Students' Perceptions on Practices for Monitoring Team Contributions

Monitoring Practice	Student Perceptions
Checking the Collaboration Platform	(+) Honest & fair indication of the team contributions (+) Effective in identifying slackers, avoiding social consequences (−) Not all types of project contributions can be captured (−) Metrics might be misleading (e.g., when pair programming) (−) Code quantity not representative of effort, varying difficulty (−) Susceptibility to manipulation/cheating Should be combined with other practices to capture context
Reporting on Team Members' Contributions	(+) Interim & final reports can give insight into contributions (+) Peer reviews as feedback from teammates (+) Anonymous reviews allow for honesty (−) Dishonesty, avoiding doing negative evaluations, lack of incentive (−) Susceptibility to team collusion, "easy to cheat" (−) Lack of transparency in final grades
Presentations or Verbal Group Assessments	(+) Useful, fair, transparent, motivating (+) Opportunity to get instructor's feedback for the project (+) Opportunity to involve shy group members to the discussion (−) Can be ineffective in recognizing slackers (−) Time-consuming, hard for nervous students
Individual Interviews	(+) Good for checking contributions and as verbal assessments Privacy might allow speaking comfortably about work distribution (−) Time-consuming and hard to realize
Instructor Splitting up Work	(+) Accountability, simplifies project setup and grading (+) Removes projects management tasks from the team (−) Negative effect on learning
Reporting on Slackers	(+) Required control mechanism Requires openness and trust to instructors for students to initiate Hurdled by friendships, social consequences, incentives, project focus

individual grades, as reported by four students (from three universities). One student described a course where the grade was quantitatively evaluated based on metrics such as the numbers of commits and lines of code. Another student experienced, instead of the instructor directly checking the repository, that metrics including lines of code, commits, comments, and merge requests had to be included in a project report.

The students mentioned several considerations, both positive and negative, on the practice of monitoring through the source code repository. The practice was considered by half of the students who discussed it as an *"honest"* (S19) and *"fair"* (S13) indication of the team contributions. It was also recognized as an efficient and effective way to identify slacking members, without the responsibility of students reporting on them, thus avoiding social consequences. At the same time, however, the students felt that the source code repository cannot provide information on contributions related to project setup, software design, project management, project documentation, reporting, presentations, inspecting, learning from each other, brainstorming, and generating ideas. Especially this last aspect was important, according to S8: *"in the end you can see who wrote what, who worked on what. But the idea might come from both people and usually it does, it's a collaboration of two people talking, and then they agree on something and it's usually better than something one person would produce."*

For information that can be captured by the source code repository, metrics might not be representative. Metrics related to the quantity of the source code committed from each team member, such as the lines of code and the number of commits, were commonly found to be misleading for several reasons. These reasons include that students may work from a common machine and commit from one account, and that code quantity is not representative of effort because, for example, writing concise, good quality code is harder than being verbose. As S2 described it, *"it's a really difficult subject, and that is the biggest flaw in the automation. You have to determine what is good and what is always good,"* and the always good might not apply to most metrics, such as lines of code. For the same reason, metrics related to code complexity were considered misleading. The lines of code metric were also found to be misleading because of the varying difficulty in the code that is being written and its susceptibility to cheating (adding less structured code or too many comments, for instance). Similarly, metrics related to numbers of commits were found easy to cheat (committing useless code or removing and adding code in subsequent commits). An even more commonly reported drawback for commit-related metrics concerns the different way in which students work with commits, with some committing larger bunches of code than others. Manual inspection of commits was considered to be useful, to check what sort of code each member is writing, and especially for commits that consist of merge requests.

Metrics for monitoring the development process were also discussed. The use of discussions and comments, for example on issues or merge requests, has the drawback that discussions are not always documented in the repositories, because *"if people sit together in a room, then they tend to have discussions face to face instead of online"* (S4). Still, some students found it useful to monitor if team members are active in the repository while the project is running *"and not by the number of commits, but by activity. You know, if a person commits only once a week, but it's always at a certain moment and it's huge, then that's fine"* (S3). This was found especially useful for identifying members that contribute too much, as well as slacking members, as S16 said, *"you can review the amount of commits and the amount of lines, but I think if you do that, it's only to see whether someone did nothing. So you can easily separate the bad apples."*

Aside from monitoring the source code repository, one student described a project where the team communications had to be through a prescribed messaging channel that was monitored by the teaching team, along with all other development activities. Although this had the disadvantage that any face-to-face communication had to be added to the communication channel as well, it was recognized that having to document all communications in the channel helped in keeping the team members accountable.

Finally, students commonly indicated that information from the collaboration platforms should be used in combination with other practices, which should include communicating with the team on contributions and imbalances. Several students stressed that the opinion of the team should not be overlooked, especially when defending a member concerning information from the repository because, as an instructor, *"you have no context about it"* (S1). The inquiry might not lead the team to give input on the contributions of slacking members because, as S19 said, when his team was called upon in this situation, *"It's not my fight to fight."*

4.3.2 Reporting on Team Members' Contributions. Eleven students from four universities described at least one experience when they had to write a report on the contributions of themselves and other team members. This concerned general reports, where *"you would have to write who did what"* (S8), reports of each member on their own contributions on the project, and, in several cases (reported by students from three universities), peer reviews on the other members of their team. These reviews were mostly secret, disclosed only to the teaching team, and they could be structured around specific criteria relating to the contributions, communication, and collaboration aspects.

Two students (from different universities) mentioned having to fill in quantified assessments of other team members, either using scales on specific criteria or assigning participation grades. Not all mentioned peer assessments were secret, though. Students from one university had experiences with writing feedback on open report fields, that could be read by their team members. S5 described that he *"would use the open ones to say, you know, you have to give everybody a tip and a top right. Did you do well? What could you improve on?"* To support this process of open and closed peer assessments, a student from the same university mentioned that dedicated reporting software was used by almost all courses with projects, where assessments are automatically sent out to be filled.

Reports are being used in various ways. In some cases, they were used as progress reports, handed in two or three times during projects, while in most cases reporting was only done at the end of projects. Concerning their goal, students from three universities described reports being used to assign individual grades. S17 described being *"graded on the basis of our contribution, and the teacher will determine our contribution by forcing us to all, at the end, fill in a form about all of the other team members contribution."*

Students reported both advantages and disadvantages of the practice of reporting on team contributions. On the one hand, it can give the instructor insight into what everyone does, with interim reporting being beneficial for intermediate checks *"that could help them identify group work problems earlier on"* (S3) and repeated peer reviews enforcing continuous monitoring that *"sparked the motivation by design"* (S18). Peer reviews were also viewed as valuable feedback in itself, with S5 saying that *"there's definitely value in a peer reviewing each other, separately from the grade, even just to give each other an idea of how you feel you are working and how you feel the others are working,"* but also because *"it's just useful to see how you are as a person and to recognize your strengths and flaws for the next project"* (S8). Anonymous or closed reviews promote, according to several students, honesty in expressing their opinions in comparison to open reporting or to one-to-one meetings with the teaching staff.

On the other hand, one drawback of team contribution reports lies in the possibility of dishonesty, especially towards avoiding doing negative evaluations. Dishonesty can occur because of fear that the author will be identified, *"he would kind of know who to get angry at for having a lower grade and if that grade means he would fail the course, then he would be angry at someone he worked with"* (S2), because of fearing to give a negative image towards the teaching team, so that as a student *"may not be saying what you really think because you would think, 'yeah, if I say the wrong thing or what I really think, then my grade will get impacted'"* (S10), or because of lying about your own contributions or to "backstab" team mates. Another drawback lies in the possibility for an entire team to be dishonest towards the teaching team, colluding towards the image that they will show. The practice was found to be *"too easy to cheat on"* (S4). Most often friendship was the reason for colluding teams, but other reasons were mentioned. These include the avoidance of negative consequences such as having to discuss the collaboration, as described by a cum laude-level student who admitted assigning high grades for the whole team when he had done the project on his own, and the lack of incentive for rating team mates negatively in anonymous reports: *"you don't really get a bonus from having more contribution. You just get a subtraction from doing less. So I don't really have an incentive for someone to get a lower grade because it doesn't affect me in any way. So I don't really care that much"* (S17). Finally, a problem identified with secret reports was the lack of transparency in the final grades. As S2 summarized it, *"it's also not nice failing someone without giving them a valid reason. Cause you can say, 'yeah, but you didn't do as much.' They can ask 'according to who' and if you can't answer that, then you're taking away the reason that they failed the course."*

4.3.3 Presentations or Verbal Group Assessments for Checking Contributions. Group presentations and verbal group assessments are used to evaluate the contributions of team members, as discussed by seven students (from two universities). These were described either as *"post mortem discussions with the entire group,"* which included a verbal assessment, or, if in front of the entire class, as group presentations or "mini defenses." As S3 explained, *"in many courses, they ask you to present your work and then one or two people present and then afterwards the whole group comes up on stage and the professor randomly asks questions and then the professor can get a very balanced view of who did what."* Students reported on various types of questions, some of which were on their own contributions, some were generated from peer reviews handed in before the presentations, and some were examination-type questions, for example on how things were implemented. Most presentations occurred at the end, and one student mentioned a course with an interim presentation.

Five students who expressed their perceptions on group presentations generally found them useful. Benefits included that this practice was considered fair and that it provides an opportunity to get feedback for the project, *"it also gives you an insight of what professors think about your project. [...] Then you get your mark based on what you present and that gives you a hint and poke in the right places"* (S8). Interim presentations were considered useful for the same reason. Further, group presentations are transparent and prohibit potential "backstabbing" that may happen in individual meetings, they can be motivating according to one student because they *"incentivized that you do well in the project because you don't want to be blamed by six professors, like roasted."* (S18), and the technical part of presentations can be a good way to involve shy group members to the discussion, because *"at some point the professor asks about a specific part of the work and then they are the ones who did the most work on that. And then they'll naturally start speaking"* (S3). Negative aspects of team meetings include that they might not be tough enough to recognize slackers, as *"some people are really smart in like presenting themselves. So for them it won't work"* (S6). Presentations were also recognized to be time-consuming, some people can get really nervous and, according to one student, they can be uncomfortable for participants and spectators when a group member is "failing."

4.3.4 Individual Interviews, Instructor Meetings, Reporting on Slackers, Splitting up Work. Several other, less frequently mentioned practices to monitor contributions came up. First, the practice of having individual interviews to assess the contributions of the team members in a group project was discussed by six students, two of whom (from different universities) described experiences with this. Overall, the perception was positive in its possibility to check contributions and as a verbal assessment. Perceptions on privacy varied, with one student considering that it allows speaking more comfortably about the work distribution when the other team members were not present, and another student saying that in a one-to-one meeting it might be difficult for a student to "speak the truth." Individual interviews were also said to be more comfortable than presentations for people who tend to get nervous in presentations, while it was recognized that they are time-consuming and hard to realize.

Second, the practice of having regular meetings with the teaching team was discussed by four students. Three of these students (from one university) described participating in a group project where their progress and contributions were monitored during such meetings. S1 described it as *"every two weeks or something we would meet with our guiding professor and we both had to talk of course. And the teacher also can kind of tell from you whether you're both doing the same thing or working as hard and adding as much value."* Two students mentioned that the grades that the team members got for the project were influenced by the perceptions that the teachers got from these meetings for the members' contributions, either by directly participating or from what the

teaching assistants documented. It was recognized by one student that having regular meetings might be hard to realize in large classrooms, but it would be useful in managing slacking members.

Third, four students discussed the possibility of the teacher splitting a group project into individual projects to be assigned to the team members. None of the students actually experienced this practice, however they suggested ways to implement it, for example, *"everyone gets a part and there is a combined part and they get like a grade and get like an average of the combined part and their own parts"* (S15). Perceptions on this practice were generally positive. Benefits included that it would make team members more accountable, that it is a simple project setup strategy, it offers an easy way of grading, and splitting into parts can guide the students towards the implementation. It was also found to simplify communications within the group by removing project management tasks from the team, because *"nobody tries to be like a leader. So either there should be a system like that where you need to pick who is responsible for the whole group [...] or the teacher could break down the assignments and [...] confirm who is going to do which part"* (S9). The single drawback that was mentioned was its negative effect on learning: *"depending on what you want to learn from the assignment, you don't get to see everything. And if you don't get to see everything you'll obviously miss a part of the assignment"* (S15). Finally, an extreme case of splitting up the workload mentioned by one student involved distributing the various assignments of a course between team members. However, *"the problem with that is that you don't have any control over the other assignments"* (S14).

Finally, six students described the practice of filing a complaint to the teacher on team members that are not contributing. This was recognized as a required control mechanism *"so that people who don't want to do anything are punished"* (S18), because they can have a large impact on the team. Some prerequisites were identified. First, openness and trust to the professors for the students to initiate this process is required. Second, complaints should be taken into account, because sometimes teachers *"say report it yourself and then don't do anything with it"* (S3). Finally, complaints should be filed before the end of the project, because *"if they finish a project and then say, 'yeah, you didn't do anything,' they should have told them earlier"* (S4). Several reasons were identified for not filing complaints on slacking team members: students might overlook it because of friendships, as *"nobody will rat their friend out"* (S13), but also to avoid the social consequences, for example other students *"holding a grudge against them."* This is especially the case when the consequences for the slackers are severe, for example because of course dependencies. S4 described a project where a slacking member was reported and *"it all shifts up an entire year if you fail that course. Um, which is kind of disastrous."* In view of the consequences, slacking might be seen as not important enough to file a complaint. As S10 explains *"I also don't dislike that so much that I would tell the teacher about it because it's, you know, then you have to tattletale on them. And I also don't wish that upon them that they would have to do it again or something."* Another student added the reason of staying focused on the project: *"I was just stressed. I just wanted to work and not deal with the outside stuff"* (S1).

5 DISCUSSION

Our study revealed various setups and grading strategies that are used in group programming assignments in computer science courses and provided insights into the advantages and disadvantages of group formation policies as perceived by students, as well as into alternative grading strategies and practices to monitor contributions. Below, these insights are discussed, together with provisional recommendations for educational practice. Although we believe that the described student perceptions provide starting points for such recommendations, it should be noted that, because these insights concern experiences from a small and diverse group, these practices should be further considered (in collaboration with the practice) before they are applied.

Examining the unique characteristics of group programming assignments, the varying levels of prior programming experience and skills of computer science students were found to affect

several aspects of group work: the team formation decisions, with large differences in skill level being mostly undesirable and high-skilled students sticking together, the learning opportunities, with very qualified members often overtaking disproportionally large shares of work, as well as the grading factors, giving rise to the underlying issue of whether it is fair if increased efficiency is rewarded with a higher grade. Further, it can be considered whether targeting the process of team formation (by matching students with similar skills) can be helpful in decreasing the issue of slacking members.

The use of metrics and process data made available by the group collaboration platform was also found to be common. Data from the source code repository was thought to be useful in monitoring activity and recognizing slacking members, especially because, from all recognized monitoring practices, this was the practice identified as lifting the weight of ratting out off the team. For this reason, we believe that using it in an unobtrusive way, for the purpose of identifying slacking members and initiating inquiries, can be beneficial. Heavier use of automated metrics should, however, be done cautiously. Most students agreed that these cannot give a full picture of the work distribution. Moreover, issues that apply to automated productivity measurements in software development teams [26] were also found to apply to student teams: They are not always representative, they should be combined with qualitative insights to explain the measured work distribution, and they can influence the motives and behavior of the students from working towards a common goal towards working to improve individual metric results.

Regarding grading strategies, our findings are in line with the literature, which is not definitive on the practice of assigning the same or individual grades for teamwork. Differentiating between group members' contributions is considered appropriate and fair [14], but it is difficult to determine the best approach to apply this [14, 21]. Furthermore, assigning the same grade to all team members is a well-recognized way to structure reward interdependence for the combined efforts of the group so they can be united around a common goal [20]. This is also one of the main advantages of this strategy highlighted during our interviews. We found, however, that this strategy is exploited for slacking, confirming earlier findings that a common group grade based on a single group product may set up conditions where one or two members do most of the work [11, 40]. The assessment and monitoring techniques that were reported by instructors in Reference [17] were also identified in our study, where students similarly indicated that peer assessment can suffer from team collusion and the reluctance to inform instructors on the contributions of teammates.

Students express various opinions on the three main grading strategies they describe. We find, however, that the identified advantages and disadvantages of each are useful when considered side-by-side with the reasons to implement a programming assignment as a group project in the first place. If it is for practical reasons of scaling to the large number of students, then the advantages of the self-selection of the team and of assigning individual grades, while enforcing several tight practices for monitoring contributions, would outweigh their disadvantages. It should be stressed that, in that case, several monitoring practices should be combined to ensure the integrity of the individual grades, as revealed both during our interviews and through other works [10, 17]. If the reasons for a group programming assignment pertain to learning through collaborating with others, or to increasing the students' collaboration skills, then having instructor-selected teams and assigning the same group grade, together with less competitive monitoring strategies such as regular group meetings or presentations, would allow the members to focus on the project, get a group feeling and self-organize, without focusing on each member's contributions. The importance of recognizing the goal of a group assignment and its consequences on required or optimal setup and support have been recognized before [25].

Independently of the applied strategies, we should stress the need for sharing beforehand the grading criteria, as well as for transparency and feedback, requirements that were recurring in the

interviews but are also well documented in the bibliography [13, 33]. It can also be valuable to be transparent to the students on the overall aim of the group assignments, clarifying for them whether and why aspects such as their collaboration skills are being evaluated as well. Finally, it can be remarked that an important underlying question is the extent to which strict monitoring of university level students' work should be desirable or necessary in the first place, given that a certain level of independence can be expected at this point in their education. Although future research could assess this aspect further, possibly by including the perspective of instructors, it can also be considered that higher transparency on the aim, setup, and grading of an assignment, as well as more guidance in group formation, can ultimately contribute to more independent working by the students.

5.1 Limitations

Several limitations of this study can be identified. First, our research was based on interviews conducted with a small number of students of four universities of one country. The experiences and perceptions of this student population may differ from the ones of other students in the same or other institutions, countries, and cultures. Additionally, students who consent to participate in interviews about group work and have their answers used in research projects may not reflect the general student population. Second, regarding the internal validity of our study, a threat is the social pressure that the respondents might have felt when disclosing their perceptions about group work and about the policies they have encountered during their studies. Overall, all students appeared comfortable during their interview and not hesitant to give their honest opinion. Still, they might have answered differently with another interviewer or in a more anonymous data-collection setting. Differences between interviews might also have affected the results: Half of the interviews were physical and others online, and some students were native speakers while others were not. Third, concerning data processing, in the case of a thematic analysis of the type of data as included in the current research, decisions on the approach are guided by the underlying aim of the study [5]. In the case of our study, the aim was to explore the perception of students, therefore, within our pre-specified topics, the approach was data-driven. The different experiences and ideas of the students were integrated yet described extensively, giving context and providing quotes to illustrate and substantiate our interpretation.

6 CONCLUSIONS

The goal of this article is to explore the experiences and perceptions of computer science students on the setup and grading of group programming assignments. Interviews with students from four universities informed our study on team formation and composition policies and their effects on group work, on group monitoring techniques, on the use of automated metrics and whether they reflect work distribution, and on the advantages and disadvantages of three specific grading strategies.

Our findings confirm the specificity of programming assignments, especially in terms of monitoring opportunities due to the available process data, grading factors that may include code quality, and issues caused by varying levels of prior programming experience with teams. Most importantly, they stress the value of an approach to the setup and grading of group assignments that is transparent, tailored to the aim of the specific course and assignment, that utilizes the available process data from the collaboration platform and, finally, that combines strategies to monitor and evaluate.

Our results suggest several possible directions for future work. The effect of the varying skills and prior programming experience, as well as of other characteristics such as gender, could be studied in depth through both qualitative and quantitative studies. The influence of such factors would

be especially suited for a large-scale quantitative approach in which a model is composed and tested, identifying the role of these factors as well as their interconnectedness. Similar larger-scale studies in other countries and educational institutions could highlight culture-specific differences in the perceptions of students. Experiments could evaluate tool support for monitoring source-code repositories for student teams. Moreover, this study focused on assignment setup, without examining aspects related to the how the student teams work in their programming projects, how they distribute work between the members, and how they solve problems, which would be promising directions for future work.

ACKNOWLEDGMENTS

The authors would like to thank Marina Milo (Vrije Universiteit Amsterdam) for helping with setting up the interviews and processing the interview transcripts. We would also like to thank the students that were involved in this research. We are grateful for your time and for sharing your experiences and insights with us.

REFERENCES

[1] Christine Alvarado, Gustavo Umbelino, and Mia Minnes. 2018. The persistent effect of pre-college computing experience on college CS course grades. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE'18)*. ACM, New York, NY, 876–881.

[2] Omar Badreddin, Wahab Hamou-Lhadj, Vahdat Abdelzad, Rahad Khandoker, and Maged Elassar. 2018. Collaborative software design and modeling in open source systems. In *System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering*, Ferhat Khendek and Reinhard Gotzhein (Eds.). Springer International Publishing, Cham, 219–228.

[3] Yongmei Bentley and Shamim Warwick. 2013. Students' experience and perceptions of group assignments. *J. Pedag. Dev.* 3, 3 (2013), 11–19.

[4] Erik Blair. 2015. A reflexive exploration of two qualitative data coding techniques. *J. Meth. Meas. Soc. Sci.* 6, 1 (2015), 14–29.

[5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualit. Res. Psychol.* 3, 2 (2006), 77–101.

[6] Kevin Buffardi. 2020. Assessing individual contributions to software engineering projects with git logs and user stories. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE'20)*. Association for Computing Machinery, New York, NY, 650–656. DOI: https://doi.org/10.1145/3328778.3366948

[7] Rachel Cardell-Oliver. 2011. How can software metrics help novice programmers? In *Proceedings of the 13th Australasian Computing Education Conference - Volume 114 (ACE'11)*. Australian Computer Society, Inc., 55–62.

[8] Kenneth J. Chapman and Stuart van Auken. 2001. Creating positive group project experiences: An examination of the role of the instructor on students' perceptions of group projects. *J. Market. Educ.* 23, 2 (2001), 117–127. DOI: https://doi.org/10.1177/0273475301232005

[9] Jian Chen, Guoyong Qiu, Liu Yuan, Li Zhang, and Gang Lu. 2011. Assessing teamwork performance in software engineering education: A case in a software engineering undergraduate course. In *Proceedings of the 18th Asia-Pacific Software Engineering Conference*. 17–24. DOI: https://doi.org/10.1109/APSEC.2011.50

[10] Nicole Clark, Pamela Davies, and Rebecca Skeers. 2005. Self and peer assessment in software engineering projects. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42 (ACE'05)*. Australian Computer Society, Inc., 91–100.

[11] Carol L. Colbeck, Susan E. Campbell, and Stefani A. Bjorklund. 2000. Grouping in the dark. *J. High. Educ.* 71, 1 (2000), 60–83. DOI: https://doi.org/10.1080/00221546.2000.11780816

[12] Michelle Craig, Phill Conrad, Dylan Lynch, Natasha Lee, and Laura Anthony. 2018. Listening to early career software developers. *J. Comput. Sci. Coll.* 33, 4 (Apr. 2018), 138–149.

[13] Susan Brown Feichtner and Elaine Actis Davis. 1984. Why some groups fail: A survey of students' experiences with learning groups. *Organiz. Behav. Teach. Rev.* 9, 4 (1984), 58–73. DOI: https://doi.org/10.1177/105256298400900409

[14] Graham Gibbs. 2009. The assessment of group work: Lessons from the literature. *Assessment Standards Knowledge Exchange*. Brooks University. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.422.8600.

[15] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. 2008. Measuring developer contribution from software repository data. In *Proceedings of the International Working Conference on Mining Software Repositories (MSR'08)*. Association for Computing Machinery, New York, NY, 129–132. DOI: https://doi.org/10.1145/1370750.1370781

[16] Greg Guest, Arwen Bunce, and Laura Johnson. 2006. How many interviews are enough? An experiment with data saturation and variability. *Field Meth.* 18, 1 (2006), 59–82.
[17] J. H. Hayes, T. C. Lethbridge, and D. Port. 2003. Evaluating individual contribution toward group software engineering projects. In *Proceedings of the 25th International Conference on Software Engineering*.622–627.
[18] Nicole Herbert. 2007. Quantitative peer assessment: Can students be objective? In *Proceedings of the 9th Australasian Conference on Computing Education - Volume 66 (ACE'07)*. Australian Computer Society, Inc., 63–71.
[19] Michael Hewner and Mark Guzdial. 2010. What game developers look for in a new graduate: Interviews and surveys at one game company. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*. Association for Computing Machinery, New York, NY, 275–279. DOI: https://doi.org/10.1145/1734263.1734359
[20] D. W. Johnson and R. T. Johnson. 1999. *Learning Together and Alone: Cooperative, Competitive, and Individualistic Learning*. Allyn and Bacon.
[21] Lucy Johnston and Lynden Miles. 2004. Assessing contributions to group assignments. *Assess. Eval. High. Educ.* 29, 6 (2004), 751–768. DOI: https://doi.org/10.1080/0260293042000227272
[22] Jihie Kim, Erin Shaw, Hao Xu, and Adarsh G. V. 2012. Assisting instructional assessment of undergraduate collaborative wiki and SVN activities. In *Proceedings of the International Conference on Educational Data Mining (EDM'12)*. 10–16.
[23] Michael S. Kirkpatrick. 2017. Student perspectives of team-based learning in a CS course: Summary of qualitative findings. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE'17)*. Association for Computing Machinery, New York, NY, 327–332. DOI: https://doi.org/10.1145/3017680.3017699
[24] Femke Kirschner, Fred Paas, and Paul A. Kirschner. 2009. Individual and group-based learning from complex cognitive tasks: Effects on retention and transfer efficiency. *Comput. Hum. Behav.* 25, 2 (2009), 306–314.
[25] Paul A. Kirschner, John Sweller, Femke Kirschner, and Jimmy Zambrano. 2018. From cognitive load theory to collaborative cognitive load theory. *Int. J. Comput.-supp. Collab. Learn.* 13, 2 (2018), 213–233.
[26] Amy Ko. 2019. *Why We Should Not Measure Productivity*. Apress, Berkeley, CA, 21–26. DOI: https://doi.org/10.1007/978-1-4842-4221-6_3
[27] Joanne P. LaBeouf, John C. Griffith, and Donna L. Roberts. 2016. Faculty and student issues with group work: What is problematic with college group assignments and why? *J. Educ. Hum. Dev.* 5, 1 (2016), 13.
[28] Joanne P. LaBeouf, John C. Griffith, and Marian C. Schultz. 2014. The value of academic group work: An examination of faculty and student perceptions. *Bus. Rev., Cambr.* 22, 1 (2014), 32–39.
[29] Richard Layton and Matthew Ohland. 2001. Peer ratings revisited: Focus on teamwork, not ability. In *Proceedings of the American Society for Engineering Education Conference*.
[30] P. L. Li, A. J. Ko, and J. Zhu. 2015. What makes a great software engineer? In *Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 700–710. DOI: https://doi.org/10.1109/ICSE.2015.335
[31] J. Lima, C. Treude, F. F. Filho, and U. Kulesza. 2015. Assessing developer contribution with repository mining-based metrics. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. 536–540.
[32] Yao Lu, Xinjun Mao, Tao Wang, Gang Yin, and Zude Li. 2019. Improving students' programming quality with the continuous inspection process: A social coding perspective. *Front. Comput. Sci.* 14 (10 2019). DOI: https://doi.org/10.1007/s11704-019-9023-2
[33] Larry K. Michaelsen and Michael Sweet. 2008. The essential elements of team-based learning. *New Dir. Teach. Learn.* 2008, 116 (2008), 7–27. DOI: https://doi.org/10.1002/tl.330
[34] Reza M. Parizi, Paola Spoletini, and Amritraj Singh. 2018. Measuring team members' contributions in software engineering projects using git-driven technology. In *Proceedings of the IEEE Frontiers in Education Conference (FIE'18)*. 1–5.
[35] Arnold L. Patton and Monica McGill. 2006. Student portfolios and software quality metrics in computer science education. *J. Comput. Sci. Coll.* 21, 4 (Apr. 2006), 42–48.
[36] Elizabeth Pfaff and Patricia Huddleston. 2003. Does it matter if I hate teamwork? What impacts student attitudes toward teamwork. *J. Market. Educ.* 25, 1 (2003), 37–45.
[37] Alex Radermacher and Gursimran Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13)*. Association for Computing Machinery, New York, NY, 525–530. DOI: https://doi.org/10.1145/2445196.2445351
[38] Christopher Scaffidi. 2018. Employers' needs for computer science, information technology and software engineering skills among new graduates. *Int. J. Comput. Sci., Eng. Inf. Technol.* 8 (2 2018), 01–12. DOI: https://doi.org/10.5121/ijcseit.2018.8101
[39] Joseph Seering, Ray Mayol, Erik Harpstead, Tianying Chen, Amy Cook, and Jessica Hammer. 2019. Peer feedback processes in the game industry. In *Proceedings of the Symposium on Computer-Human Interaction in Play (CHI PLAY'19)*. Association for Computing Machinery, New York, NY, 427–438. DOI: https://doi.org/10.1145/3311350.3347176

[40] Robert E. Slavin. 1983. When does cooperative learning increase student achievement?*Psychol. Bull.* 94, 3 (1983), 429.
[41] Anya Tafliovich, Andrew Petersen, and Jennifer Campbell. 2016. Evaluating student teams: Do educators know what students think? In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE'16)*. Association for Computing Machinery, New York, NY, 181–186. DOI: https://doi.org/10.1145/2839509.2844647
[42] Hamid Tarmazdi, Rebecca Vivian, Claudia Szabo, Katrina Falkner, and Nickolas Falkner. 2015. Using learning analytics to visualise computer science teamwork. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'15)*. Association for Computing Machinery, New York, NY, 165–170. DOI: https://doi.org/10.1145/2729094.2742613
[43] Stephanie D. Teasley and Jeremy Roschelle. 1993. Constructing a joint problem space: The computer as a tool for sharing knowledge. In *Computers as Cognitive Tools*, S. P. Lajoie and S. J. Derry (Eds.). 229–258, Lawrence Erlbaum Associates, Inc.
[44] Ville Tirronen and Ville Isomöttönen. 2011. Making teaching of programming learning-oriented and learner-directed. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling'11)*. Association for Computing Machinery, New York, NY, 60–65. DOI: https://doi.org/10.1145/2094131.2094143
[45] Sander Valstar, Caroline Sih, Sophia Krause-Levy, Leo Porter, and William G. Griswold. 2020. A quantitative study of faculty views on the goals of an undergraduate CS program and preparing students for industry. In *Proceedings of the ACM Conference on International Computing Education Research (ICER'20)*. Association for Computing Machinery, New York, NY, 113–123. DOI: https://doi.org/10.1145/3372782.3406277
[46] Natascha van Hattum-Janssen. 2013. Student perceptions of group work. *Univ. Minho, Res. Centre Educ. 4710* 57 (2013).
[47] 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 (SIGCSE'18)*. ACM, New York, NY, 80–85.
[48] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. 2015. The emergence of github as a collaborative platform for education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'15)*. Association for Computing Machinery, New York, NY, 1906–1917. DOI: https://doi.org/10.1145/2675133.2675284

Received July 2020; revised December 2020; accepted December 2020