Evaluating Group Work in (too) Large CS Classes with (too) Few Resources: An Experience Report

Joël Porquet-Lupine
jporquet@ucdavis.edu
Department of Computer Science
University of California, Davis
Davis, California, USA

Madison Brigham
mlbrigham@ucdavis.edu
Department of Computer Science
University of California, Davis
Davis, California, USA

ABSTRACT

Group work is an excellent way to provide students with more complex, engaging class projects and help them practice many of the professional skills necessary for industry, where large-scale projects have long been the norm. However, as instructors of large CS classes are typically unable to determine individual contributions based on project submissions alone, group work can often cause some frustration when partners of the same group don't put in the same amount of effort yet receive the same score.

In this paper, we describe a bimodal assessment strategy for group work, which couples end-of-term staff-led oral interviews with per-project student-reported self and peer evaluations. The combination of these approaches ensures a thorough and fair evaluation of students' contributions to their groups, and scales up to large classes with limited instructional staff. The feedback from students is very positive, both in terms of agreeing with the narratives justifying this assessment strategy and finding it to be an effective solution to fairly grading group work.

CCS CONCEPTS

• **Social and professional topics → Student assessment**.

KEYWORDS

Teamwork and collaboration; Undergraduate education; Large classes; Oral assessment; Self and peer assessment

ACM Reference Format:
Joël Porquet-Lupine and Madison Brigham. 2023. Evaluating Group Work in (too) Large CS Classes with (too) Few Resources: An Experience Report. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569788

1 INTRODUCTION

Group work in CS courses is known to provide numerous benefits [2, 8]. In the short term, it typically enables students to work on more complex and interesting class projects than if they worked alone. It also helps students refine their understanding of the course material, as they are incentivized to communicate about it with other students in their respective group. In the long run, group work is an indispensable way for students to prepare for their careers in industry, where large-scale projects have long been the norm. Students should therefore practice the various required professional *(soft)* skills, such as the communication/confrontation of ideas with others, the optimal allocation of tasks between group members, and the time management needed to meet deadlines. Group work also has valuable benefits for instructors of large classes without many instructional resources (e.g., few teaching assistants). If students work in groups, they can help each other more efficiently, thus indirectly reducing the need for staff-provided support (e.g., office hours, online class forum). Group work also translates directly into fewer project submissions to grade, which leaves staff more time to improve the quality of their grading.

Group work, however, comes with its share of challenges [3, 5, 7, 8, 11, 12]. From our five-year experience of running two-student group projects in 11 offerings of the same CS course, we identified three major pain points related to group work: the logistics of finding a partner, the cooperation process during the implementation of the project, and finally, the fair grading of the project, that is, taking into account each group member's contribution. The first issue usually draws only minor complaints from students and can largely be solved by using light prevention techniques, such as providing a (virtual) venue where students can meet and form groups, and imposing an early deadline by which all students must be in a group. The second and third issues are intimately linked in the sense that if a group was dysfunctional, then the student who feels wronged (often due to their partner not doing their fair share of work) will at least expect that the project will be graded accordingly. Actively monitoring group dynamics during the relatively short span of a class project is out of reach in large classes with few instructional staff, but instructors can mitigate the issue of dysfunctional groups by preemptively offering guidance on ways to collaborate effectively. In our CS course, we found that the issue of fair grading was ultimately the most difficult to address and incidentally garnered the highest number of complaints from students, the most common frustration being along the lines of *"My partner didn't do anything, yet we got the same grade."* Since project submissions don't include reliable evidence of individual contributions, the fair grading of group members has to involve some active intervention from the instructional staff.

This paper presents a bimodal assessment strategy for group work, which combines oral interviews and self and peer evaluations, and scales to large classes with few instructional resources. Oral interviews are individual assessments conducted by the instructional staff at the end of the term, and are meant to gauge each student's understanding of a selection of their group project submissions.

SIGCSE '23, March 15–18, 2023, Toronto, ON, Canada.
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569788

The self and peer evaluations are completed by students directly after each project, and are meant to provide insight into each partner's engagement and contribution to the given project. While both partners in each group receive the same grade for their project submission, our bimodal assessment strategy enables additional, individualized grades. We surveyed students over three offerings of our course that used this assessment strategy, and they largely agreed with its narratives and overall effectiveness.

This paper is organized as follows. Section 2 discusses related work on group work, oral assessments, and peer evaluations. Section 3 presents the class settings in which our bimodal assessment strategy was used. In Sections 4 and 5, we detail the two components of our assessment strategy: oral interviews and self and peer evaluations. We discuss the results from our student survey in Section 6. Finally, we conclude and suggest potential next steps in Section 7.

2 RELATED WORK

The adoption of group work in CS classes appears to be as old as the inauguration of CS classes themselves. The ACM "Curriculum 68: Recommendations for academic programs in computer science" [1] already mentioned group work in laboratory classrooms, and publications in early ACM SIGCSE conferences [2] advocated for team projects as an essential part of any CS curriculum at the undergraduate level. It is generally acknowledged that group work can be a powerful tool for practicing collaborative and communication skills, also known as professional skills, which are indispensable for students to have acquired by the time they graduate. Oakley et al.[8] even found statistically significant correlations between agreement that a course had fulfilled its objectives and the use of student teams in that course. However, as Waite et al.[14] and Koppe et al.[5] argue, simply introducing group work is often insufficient and must be accompanied by active strategies to foster a collaborative atmosphere among students.

Another measure that allows students to practice communication skills is oral assessment. Ohmann [9] and Sabin [10] successfully used oral assessment to replace typical written exams, usually for their class final exams. Sabin reports that students found the personalized, interactive nature of the exam helpful in advancing their learning and communication skills. On a wider scale, Stratton [13] argues that oral presentations create individual accountability for understanding homework solutions, balancing a typical deficiency of group work while increasing faculty-student interaction. Stratton also mentions that oral presentations are good practice for situations beyond the classroom, such as technical interviews.

A frequent student concern surrounding group work is the fair grading of each team member according to their perceived contributions. While oral assessment can help mitigate "freeloading" by penalizing students who appear not to have contributed much to their group projects, it is also possible to hear from group members directly via peer assessment. LeJeune [6] presents a comprehensive assessment approach for grading individuals in CS group projects; however, it seems to be limited to relatively small classes. The approach includes peer and instructor assessments, and factors in the quality of the contribution in addition to its mere amount. Peer assessment is, however, not broadly accepted as an adequate approach. Kennedy [4], for example, raises several potential issues, such as the reluctance of students to assess their teammates, which may affect the validity and reliability of the collected data.

3 COURSE SETTINGS

This section presents the Operating Systems course (abbreviated as CSOS hereafter) for which our bimodal assessment strategy for evaluating group work was developed.

CSOS is a concept-focused and programming-heavy course that is required for all students majoring in CS. Students generally take it at the end of their junior year or beginning of their senior year. CSOS is offered two to three times a year (our institution is on the quarter system) and enrolls around 200 students per class. In terms of instructional staff, we typically receive three teaching assistants (TAs), each with a 20-hour/week appointment.

Theory		**Practice**		**Total**
Midterm exam	Final exam	Three group projects (equally weighted)	Group work (oral interview + self and peer evaluations)	
20%	30%	40%	10% (uncapped)	100%

Table 1: CSOS course grading

As shown in Table 1, the grading for this course covers two main aspects, theory and practice, each counting for 50% of the students' final course grades. The theory portion is assessed via written examinations, typically a midterm exam and a final exam. These examinations mostly focus on materials seen in class (i.e., lectures and discussions). The practice portion focuses on three challenging group projects, each lasting about two weeks, for which students work in pairs. Students are not allowed to have the same partner more than twice, so by the end of the term, students will have worked with either two or three different project partners. The practice grade is further broken down into two parts.

For the project submissions, students of the same group receive the same score. Project submissions are partially auto-graded via a script running various tests, but are also manually reviewed by the TAs. Manual reviews are designed to give personalized feedback on the submitted code via inline comments, and assess aspects of each submission that cannot be auto-graded, such as the quality of the implementation, the coding style, etc.

On the other hand, oral interviews and self and peer evaluations are individualized scores, weighted equally. It is important to note that this individualized part of the practice grade is not capped. As explained in section 5, students can receive scores over 100% for the self and peer evaluations if it is reported that they consistently provided more work than their partners. In this case, the "overflow" is meant to indirectly adjust their (non-individualized) project submission scores. This stems from the idea that if students had to provide additional work because of their partners' lack of contributions, it is only fair that they would receive a compensating boost.

While we have taught CSOS 11 times since 2017, this paper primarily focuses on the four most recent offerings of the course (winter quarter 2020, fall quarter 2020, winter quarter 2021, and winter quarter 2022 –abbreviated as WQ20, FQ20, WQ21, and WQ22 for the remainder of this paper) since they share almost exactly the same assessment structure.

4 ORAL INTERVIEWS

This section describes the evolution of our oral interview assessment, the first part of our bimodal assessment strategy.

The main purpose of these oral interviews is to assess whether students are familiar with their project submissions, including parts that their partner may have written, as well as the underlying concepts studied in class and applied in the class projects. The idea is to mimic industry practices, where software developers are typically expected to fully understand a large codebase, even parts for which they are not the primary developers. Another, more indirect, purpose of oral interviews is to hold students who did not contribute much to their project submissions and therefore cannot explain the submitted code accountable, since a poor oral interview will negatively affect their practice grades for the class.

We started using oral interviews right from our first offering of CSOS, in 2017. At the time, interviews were performed after every group project during a 10-minute session, and each group of students was interviewed together but scored individually. This approach was extremely time-consuming, and interviewing students with their group prevented us from accurately assessing them separately. It was difficult to test students on parts of the code their partner had implemented, since their partner would often try to answer on their behalf. Furthermore, the scoring rubric employed at the time was rudimentary and only had one loosely-defined criterion. As a result, this approach wasn't satisfactory; almost all the students scored 90% or more, which was not in line with the other forms of assessment used in the class, and we would still hear a lot of frustrations from students about the fair grading of their group projects.

In its current form, oral interviews are conducted only once, during the last week of the term and following the third and last group project. This last project is completely auto-graded to alleviate the TAs' workload. Students are interviewed individually during a 10-minute session with a TA. The available time slots are announced in advance and are opened for registration at the same time for everyone. During their oral interview, each student is quizzed on two of their three group projects: one that they choose at the time of their registration, and one picked by the TA during the interview using a random number generator. Students are typically asked one or two main questions per project, taken out of a shared bank of questions written by the instructor, as well as potential follow-up questions depending on the student's answers. Within the 10 minutes of the session, we account for one minute of preparation time, about eight minutes of active interview, and one minute at the end for the TA to finalize their review. This includes selecting the adequate items in the scoring rubric and writing a few sentences about each student's performance. For a 200-student class and three 20-hour/week TAs, each TA has to interview about 67 students, therefore representing about 12 hours worth of oral interviews and thus staying well within their workload.

The improved scoring rubric consists of three criteria, covering different aspects of the interview. An excerpt is shown in Table 2. The first criterion, which accounts for 32% of the total score, aims to measure the student's overall understanding of the selected projects' concepts. The second criterion, which accounts for 40% of the score, assesses the student's understanding of the code they submitted. Finally, the last criterion, which accounts for 24% of the score, addresses the student's communication and how they formulate their answers during the interview. (Note that the missing 4% is from submitting a file on our grading platform.) Each criterion is evaluated on a 5-level scale, corresponding to different performance achievements: exemplary (100% of criterion points), accomplished (~87.5%), developing (~75%), beginning (~62.5%), and unsatisfactory (~50%). Students who don't attend their oral interview receive 0%. The interview process and the complete rubric are shared with students a few days before oral interviews are scheduled to start to help them prepare.

Figure 1 presents the grade distributions for the last five offerings of CSOS (using a standard letter grade scale for better visual clarity). They show that before implementing the current version of oral interviews (e.g., fall quarter 2019), most students would receive 100%, and almost all would score at least 90%. With the current implementation of oral interviews in the last four offerings of the course, we notice that scores are now well-distributed across the grading scale and therefore offer a much richer representation of student performance.

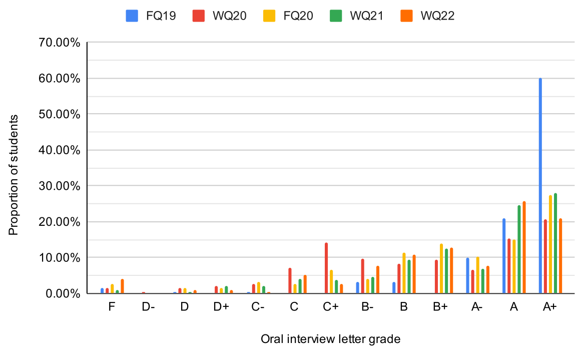

Figure 1: Oral interview grades across multiple terms

5 SELF AND PEER EVALUATIONS

In this section, we present the second part of our bimodal assessment strategy for evaluating group work: self and peer evaluations.

After each project deadline, students are given about a week to complete an evaluation survey composed of three sections. This survey aims to measure two facets of the students' group work experiences: qualitative aspects for the first two sections and quantitative for the last.

The two qualitative sections consist of Likert-scale questions measuring the students' engagement in the project, via a comprehensive list of professional and technical skills that students are expected to exhibit while working on a given project with their partner:

Understanding of concepts		Understanding of code		Communication	
+32 pts	Exemplary level (A+ – A) Understanding of the concepts is **excellent**. • Concepts covered by projects are perfectly understood and can effortlessly be explained.	+40 pts	Exemplary level (A+ – A) Understanding of the code is **excellent**. • Submitted code is perfectly understood and can effortlessly be explained (even parts that the student didn't write themselves). Student barely looks at code, only to remember very specific details if necessary. • Student is very comfortable talking about the high-level design of the code as well as the code's internals. • Student understands the limitations of their code and can discuss them.	+24 pts	Exemplary level (A+ – A): Communication is **excellent**. • Content: – Accurate, thorough, and directly on point. – Central ideas/purposes vividly stated and supported. – Well-structured sentences in effective sequence. • Verbal delivery: – Free of errors in grammar – Word choices help with clarity. – Varied and dynamic, enhanced by speech rate, volume, and tone. • Non-verbal delivery: – Eye contact. – Good posture and appropriate attire. – Helpful gestures to support presentation.
+28 pts	Accomplished level (A – B) Understanding of the concepts suffers from **a couple of important flaws**. • Most of the concepts are well understood, but difficulty to understand/explain a couple.	+36 pts	Accomplished level (A – B) Understanding of the code suffers from **a couple of important flaws**. • Student is very comfortable with most parts of the code, but has minor difficulties explaining a couple other parts. Typically needs to refer back to the code. • Student understands well the high-level design of the code, but has minor difficulty understanding a couple of the code details (or vice-versa). • Student has minor difficulties to see and discuss a couple of limitations of their code.	+21 pts	Accomplished level (A – B) Communication is suffers from **a couple of important flaws**. • Content may suffer from occasional and minor lack of focus, structure, or support. • Verbal delivery may suffer from occasional and minor errors in grammar or word choices. • Non-verbal delivery may suffer from occasional and minor deficiencies.
+24 pts	Developing level (B – C) Understanding of the concepts suffers from **a few important flaws**. ...	+32 pts	Developing level (B – C) Understanding of the code suffers from **a few important flaws**. ...	+18 pts	Developing level (B – C) Communication is suffers from **a few important flaws**. ...

Table 2: Excerpt of scoring rubric for oral interviews

- Organization: *"You had a role in the clerical organization of your group. For example, you helped define the terms of your collaboration: how/when/where you should meet, how you should work together, etc."*
- Communication: *"You had a role in the communication of your group. For example, you helped maintain constant communication with your partner throughout the project."*
- Cooperation: *"You were willing to listen and respect the ideas of your partner, and discuss the work distribution. For example, you would not try to always impose your way of doing things."*
- Attitude: *"You showed a positive and enthusiastic attitude, and it was pleasant to work with you."*
- Contribution of ideas: *"You contributed ideas to the project in terms of how to tackle the assignment, structure the code, build certain algorithms, etc."*
- Contribution of code: *"You participated in the programming aspect of the project."*

Students respond to each question on a 4-point scale, ranging from Strongly agree to Strongly disagree. A neutral option is excluded to prevent students from answering passively. In the first section of the evaluation survey, students have to assess their own engagement within the group project, while in the second section, they assess their partner's engagement. This order is purposeful, as we want students to take the time to reflect on their own engagement before they evaluate their partner's.

The last section of the survey asks students to give a rough quantitative estimate of each partner's contribution to the project. There are only five options, as the goal of this section is to capture pronounced trends in the work balance of each group, rather than an exact measure:

- 0% — 100%: *"Your partner did (almost) everything while you did (almost) nothing"*
- 25% — 75%: *"Your partner contributed substantially more than you"*
- 50% — 50%: *"You and your partner contributed (almost) equally"*
- 75% — 25%: *"You contributed substantially more than your partner"*
- 100% — 0%: *"You did (almost) everything while your partner did (almost) nothing"*

For the purpose of scoring, each option in this quantitative section is transformed into a linear score numbered from 0 to 1, by steps of 0.25.

If a student does not complete their evaluation survey after a project, one is automatically filled out on their behalf. For the two qualitative sections, their engagement is automatically set to the lowest scores, while their partner's engagement is copied directly from their partner's own self-reporting (as to avoid triggering any positive or negative deviations –see below). For the quantitative section, their contribution is derived from what their partner has reported. If their partner didn't fill out their evaluation survey either, the contribution is automatically set to 0.5 for both students.

At the end of each project evaluation, we have a total of 14 data points per student: 12 data points for engagement (six from the student via their self-evaluation and six from their partner via their peer evaluation) and two data points for contribution (similarly, one from the student and one from their partner). We average these pairs of data points and also determine two deviation scores: one for engagement and one for contribution. A deviation score is meant to capture the potential variation between what a student reported in their self-evaluation versus what their partner reported about them.

A positive deviation implies that a student may have inflated their self-reporting, while a negative deviation indicates that a student may have underestimated it. One can view deviation scores as an "inverted trust factor".

At the end of the term, we average pairs of data points across all projects. This leaves us with an average engagement score and an average contribution score. The deviation scores across all projects are also averaged, giving two deviation scores which are respectively used to adjust the engagement and the contribution scores of each student. A positive deviation is directly subtracted from the corresponding average to penalize students who inflated their self-reported performance. A negative deviation is first divided by two, and its absolute value is added to the corresponding average, therefore providing a reasonable boost to students who underestimated their self-reported performance.

The adjusted engagement and contribution scores are finally combined to produce an overall group work score, with the engagement score counting for 40% and the contribution score counting for 60%. These group work scores are not capped at 100%, especially since contribution scores can be over 0.5 (which represents a full mark for contribution) if it was reported that a student had contributed more than their partners. The idea is to reward students who provided more than their expected share on some projects.

Table 3 shows a complete example for a fictitious student. As you can see, this student slightly under-estimated their engagement for Project #1 (deviation of -0.33) and largely over-estimated it for Project #2 (deviation of 1.33), compared to what their partner(s) reported. In terms of actual contribution, they contributed as much as their partner for Project #1, contributed more than their partner for Project #2 (in agreement with what their partner reported), and underestimated their contribution for Project #3. Based on these data points, it appears that this student contributed more than their expected share across all projects, which is why their final group work grade adds up to over 100%, thus giving them a small boost for their practice grade.

The main purpose of these evaluations is to offer students the opportunity to provide meaningful feedback regarding the dynamic of their group, and hold both themselves and their partner accountable. As with oral interviews, the idea comes from existing industry practices, where regular self and peer reviews are common. Multiple times during the term, we strongly encourage students to talk to their partner before completing their evaluations, especially in situations where they may not have contributed equally, in order to avoid the negative effects of a positive deviation.

When looking at statistics from the four most recent course offerings that have adopted this group work evaluation method, displayed in Table 4, we generally observe that students are engaged in their group work and contribute equally to projects. Deviations are rather small, which suggests students are not trying to game the system. Interestingly, deviations have steadily decreased from one course offering to the next.

Once the staff sets up a generic self and peer evaluation survey, collecting evaluations from students throughout the term is straightforward. Then, using properly formatted spreadsheets, processing the evaluations at the end of the term may take a couple of hours, but it easily scales to any number of students.

6 STUDENT SURVEY

In order to collect student input about this bimodal assessment strategy used in our CSOS course, we developed a short survey[1]. The survey was set to be completely anonymous and was released in November 2021 to the 590 students who had taken CSOS in WQ20, FQ20, and WQ21. While we had 98 total respondents begin the survey (i.e., 16% of the contacted students), our questions (all optional) received between 77 and 90 answers.

In our survey, we wanted to mainly clarify two aspects: whether students agreed with our narratives around group work and our bimodal assessment strategy (Q1, Q2, Q4), and whether they agreed with our strategy's actual effectiveness (Q3, Q5). The five survey questions are listed below.

Q1. I agree with the narrative that group work is unavoidable in CS. *[narrative]*

Q2. Oral interviews mimic how you may be held accountable in the workplace (e.g., having to explain work that your entire team will have produced). *[narrative]*

Q3. Oral interviews are an effective solution to fairly grading group work. *[effectiveness]*

Q4. Group work evaluations mimic how coworkers may provide periodic reviews of themselves and one another in the workplace. *[narrative]*

Q5. Group work evaluations are an effective solution to fairly grading group work. *[effectiveness]*

Students selected their responses to each question from a 4-point Likert scale, ranging from Strongly agree to Strongly disagree. Once again, a neutral point was not offered to encourage students to make an active choice in providing either positive or negative feedback.

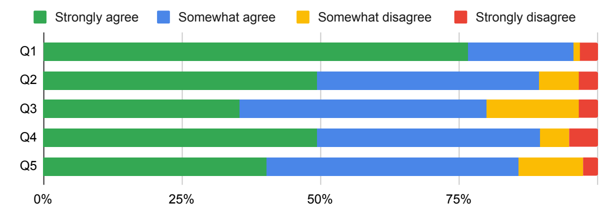

Figure 2: Student survey results

As we can see from Figure 2, almost all students strongly agree with the underlying premise of this study, in that group work is and will continue to be an integral part of their CS education and career (Q1). For our bimodal assessment strategy, students largely agree with both justification narratives (Q2 and Q4) and the claims that our assessment strategy represents effective solutions to fairly grading group work (Q3 and Q5).

7 CONCLUSION

In this paper, we described a bimodal assessment strategy for group work, which couples end-of-term staff-led oral interviews with per-project student-reported self and peer evaluations. The combination of these approaches ensures a thorough and fair evaluation

[1]Note that this study's protocol, which included the student survey, were examined by our Institutional Review Board and determined not to require a review.

	Engagement (40%)								**Contribution (60%)**	
	Organization	Communication	Cooperation	Attitude	Contribution of ideas	Contribution of code	Engagement average	Engagement deviation	Project contribution	Contribution deviation
Project #1: Self evaluation	4	4	4	4	3	3	3.66	3.66 - 4 = -0.33	0.5	0.5 - 0.5 = 0
Project #1: Peer evaluation	4	4	4	4	4	4	4		0.5	
Project #2: Self evaluation	4	4	4	4	4	4	4	4 - 2.66 = 1.33	0.75	0.75 - 0.75 = 0
Project #2: Peer evaluation	3	2	2	2	3	4	2.66		0.75	
Project #3: Self evaluation	4	4	4	4	4	4	4	4 - 4 = 0	0.5	0.5 - 0.75 = -0.25
Project #3: Peer evaluation	4	4	4	4	4	4	4		0.75	
Raw averages							3.72	0.33	0.625	-0.0833
Adjusted scores								3.72 - 0.33 = 3.39 3.39/4*100 = 84.74%		0.625 + (0.0833/2) = 0.66665 0.66665/0.5*100 = 133.33%
Grade										84.74*0.4 + 133.33*0.6 = 113.90%

Table 3: Self and peer evaluations grading example for fictitious student

	Engagement			Contribution			Final Score Average
Term	Score (out of 4)	Deviation	Adjusted Score	Score (out of 1)	Deviation	Adjusted Score	Grade
WQ20	3.61	0.05	88.01	0.49	0.03	89.90	89.15
FQ20	3.60	0.04	87.95	0.48	0.02	89.69	88.99
WQ21	3.62	0.03	88.71	0.48	0.01	94.02	91.90
WQ22	3.62	-0.02	89.06	0.48	0.01	92.37	91.05
Average	3.61	0.03	88.43	0.48	0.02	91.50	90.27

Table 4: Self and peer evaluation statistics across multiple terms

of students' contributions to their groups, and scales up to large classes with limited instructional staff. The feedback from students is very positive, both in terms of agreeing with the narratives justifying this assessment strategy and finding it to be an effective solution to fairly grading group work.

As instructors, we observed a noticeable decrease in the number of complaints from students about the fairness of group work grading. It seems like our per-project self and peer evaluations do provide a welcomed outlet for those who want to hold a non-contributing partner accountable. However, this strategy does not solve the problem of dysfunctional groups during the span of a project, which now remains the major source of frustrations for the affected students. Unfortunately, this issue is incredibly challenging to address since actively monitoring dozens of groups in large classes with few instructional staff doesn't appear within reach.

A potential next step we would like to explore regarding self and peer evaluations is the possibility of computing an "impact score" for each student, based on their cumulative deviations. This could be used in a feedback loop to post-process the impact of their scoring on their partners' group work grade. It unfortunately happens that some students are both uncooperative and dishonest, so if a student's impact score is low, then their self and peer evaluations would be given less weight compared to what their partners reported.

Group work in CS classes will likely never be completely fair, but we believe that the presented strategy helps mitigate some of its most prominent challenges.

ACKNOWLEDGEMENT

We thank Prof. Christopher Nitta for his valuable feedback on this paper.

REFERENCES

[1] William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, and David M. Young. 1968. Curriculum 68: Recommendations for Academic Programs in Computer Science: A Report of the ACM Curriculum Committee on Computer Science. *Commun. ACM* 11, 3 (mar 1968), 151–197. https://doi.org/10.1145/362929.362976

[2] John H. Crenshaw. 1978. Team Projects in the Undergraduate Curriculum. In *Papers of the SIGCSE/CSA Technical Symposium on Computer Science Education* (Detroit, Michigan) *(SIGCSE '78)*. Association for Computing Machinery, New York, NY, USA, 203–205. https://doi.org/10.1145/990555.990625

[3] Helen Drury, Judy Kay, and Warren Losberg. 2003. Student Satisfaction with Groupwork in Undergraduate Computer Science: Do Things Get Better?. In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20* (Adelaide, Australia) *(ACE '03)*. Australian Computer Society, Inc., AUS, 77–85.

[4] Geoffrey J. Kennedy. 2005. Peer-Assessment in Group Projects: Is It Worth It?. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42* (Newcastle, New South Wales, Australia) *(ACE '05)*. Australian Computer Society, Inc., AUS, 59–65.

[5] Christian Köppe, Marko van Eekelen, and Stijn Hoppenbrouwers. 2015. Improving Student Group Work with Collaboration Patterns: A Case Study. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2* (Florence, Italy) *(ICSE '15)*. IEEE Press, 303–306.

[6] Noel LeJeune. 2006. Assessment of Individuals on CS Group Projects. *J. Comput. Sci. Coll.* 22, 1 (oct 2006), 231–237.

[7] Torben Lorenzen, John Santore, David Glassman, and Juozas Baltikauskas. 2007. No Slacker on Team Programming Projects. *SIGCSE Bull.* 39, 4 (dec 2007), 117–118. https://doi.org/10.1145/1345375.1345428

[8] Barbara A. Oakley, Darrin M. Hanna, Zenon Kuzmyn, and Richard M. Felder. 2007. Best Practices Involving Teamwork in the Classroom: Results From a Survey of 6435 Engineering Student Respondents. *IEEE Transactions on Education* 50, 3 (2007), 266–272. https://doi.org/10.1109/TE.2007.901982

[9] Peter Ohmann. 2019. An Assessment of Oral Exams in Introductory CS. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 613–619. https://doi.org/10.1145/3287324.3287489

[10] Mihaela Sabin, Karen H. Jin, and Adrienne Smith. 2021. Oral Exams in Shift to Remote Learning. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 666–672. https://doi.org/10.1145/3408877.3432511

[11] Thomas J. Scott, Ralph B. Bisland, Lee H. Tichenor, and James H. Cross. 1994. Handling Interpersonal Issues for Student Team Projects. In *Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education* (Phoenix, Arizona, USA) *(SIGCSE '94)*. Association for Computing Machinery, New York, NY, USA, 397–398. https://doi.org/10.1145/191029.191205

[12] Harold H. Smith and Debra L. Smarkusky. 2005. Competency Matrices for Peer Assessment of Individuals in Team Projects. In *Proceedings of the 6th Conference on Information Technology Education* (Newark, NJ, USA) *(SIGITE '05)*. Association for Computing Machinery, New York, NY, USA, 155–162. https://doi.org/10.1145/1095714.1095751

[13] John A. Stratton. 2021. Enhancing Faculty-Student Interaction in an Undergraduate Algorithms Course Through Group Oral Presentations. In *Computing Education Practice 2021* (Durham, United Kingdom) *(CEP '21)*. Association for Computing Machinery, New York, NY, USA, 25–28. https://doi.org/10.1145/3437914.3437975

[14] William M. Waite, Michele H. Jackson, Amer Diwan, and Paul M. Leonardi. 2004. Student Culture vs Group Work in Computer Science. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA) *(SIGCSE '04)*. Association for Computing Machinery, New York, NY, USA, 12–16. https://doi.org/10.1145/971300.971308