Evaluating Individual Contribution Toward Group Software Engineering Projects

Jane Huffman Hayes
University of Kentucky
hayes@cs.uky.edu

Timothy C. Lethbridge
University of Ottawa
tcl@site.uottawa.ca

Daniel Port
University of Southern California
dport@almaak.usc.edu

Abstract

It is widely acknowledged that group or team projects are a staple of undergraduate and graduate software engineering courses. Such projects provide students with experiences that better prepare them for their careers, so teamwork is often required or strongly encouraged by accreditation agencies. While there are a multitude of educational benefits of group projects, they also pose considerable challenge in fairly and accurately discerning individual contribution for evaluation purposes. Issues, approaches, and best practices for evaluating individual contribution are presented from the perspectives of the University of Kentucky, University of Ottawa, University of Southern California, and others.

The techniques utilized within a particular course generally are a mix of (1) the group mark is everybody's mark, (2) everybody reports what they personally did, (3) other group members report the relative contributions of other group members, (4) pop quizzes on project details, and (5) cross-validating with the results of individual work.

1. Introduction

Group or team projects are a staple of undergraduate and graduate software engineering courses. Such projects provide students with realistic experiences that better prepare them for their careers. This is so widely acknowledged as important that it is often required or strongly encouraged by accreditation agencies.

In this paper, we will look at some of the issues surrounding assessment of group work, since both instructors and students complain of the difficulties of doing this in a manner that is both fair and promotes educational objectives.

The authors of this paper come from several different universities each of which has a variety of courses involving group projects. The University of Kentucky requires group projects in the CS 499 Senior Design Project course and CS 616 Software Engineering course; the University of Ottawa assigns a group full-year capstone project (SEG 4000) [4] as well as group projects in several other courses; and the University of Southern California assigns group projects both in its one semester undergraduate software engineering course CS477 and its full year graduate level course CS577.

We note that although the project characteristics and grading criteria vary in each institution and course, the problems in grading group projects seem universal. Specifically, it is very difficult to objectively determine an individual student's contribution to a group project. Various approaches have been implemented with varying degrees of success.

The paper is organized as follows: Section 2 presents background information on group projects and challenges in evaluating these for grades, Section 3 presents approaches to grading individual effort, Section 4 presents grading practices that the three universities feel worked well, and Section 5 presents a summary and conclusions.

2. Background

In this section, we discuss the characteristics and benefits of group projects as well as the challenges of grading such projects.

2.1 Characteristics/benefits of group projects

The group project has become a staple of software engineering courses at the graduate and undergraduate level. Students are usually grouped into teams of three or more and are required to perform the entire software development lifecycle in one or more semesters. The project may be "real" with a customer and/or end user who will use the software product after its completion. Generally, the teams are required to prepare artifacts throughout the lifecycle, such as management plans, concept documents, requirement specifications, design specifications, test plans, etc. The end result of the project is often a documented software product. Milestone reviews may be held throughout the course to allow the instructor to determine progress. Also, an acceptance test may be required to assure successful completion of the project.

There are many benefits to group projects: They encourage students to learn to work in groups (stressing co-operation, teamwork, and negotiation); projects enable better learning by having students learn from each other, motivate each other, rely on each other, have to work at agreed-upon times, etc. Projects enable students to develop larger or more complex systems than would otherwise be possible, due to division of labor; projects also allow students to gain experience working as part of a team, as is normally the case in industry.

Unfortunately, grading team projects is not an easy undertaking, and has many inherent challenges. Some of these, such as being fair, consistent and encouraging, apply to all forms of grading and are discussed in the next section. Group projects add additional challenges: the grading scheme must accurately reflect each individual's contribution to the overall project, and must reflect the difficulty of the project (if each group is allowed to select their own project). Also there may need to be a mechanism to account for situations where students drop out of groups or are 'fired.'

2.2 Existing literature

Literature on this topic has been relatively sparse; the following summarizes some relevant work.

The question of how to manage group projects has been studied in the field of education theory called "co-operative learning". This is discussed in some depth by Speck [7]; he covers grading group work, among other issues, but doesn't give many useful guidelines.

Johnson et al [3] propose several essential factors that must be present for effective co-operative learning in group projects: Firstly, there must be "positive interdependence" among the team members – i.e. the group members must benefit from each others' presence. Secondly, the group members must possess and use effective inter-personal skills. Thirdly, and of key relevance to this paper, the group members must hold each other *accountable* for their share of the work and analyze as a group how effectively they are co-operating.

Schultz [6] performed a survey in which he asked teams of students doing engineering projects what their greatest difficulties were with respect to teamwork. 31% of them cited "motivating slackers" as a concern; Schultz reports that many students felt "a lot of bitterness" around this issue.

The notion of accountability also appears in the work of Gates et al [2]. They focus on motivating students to contribute equally, so that the need for differentially evaluating their contributions is lessened. They suggest two strategies: First, students should maintain rough drafts of all work they contributed, including emails etc. The intent of this is to provide a body of evidence that students can use to back up their claim to having contributed effectively. Second, each group produces a statement describing what each person did in detail. Each member of the team must then sign that they agree with the statements made by each member.

McKinney [5] has produced a well-referenced set of guidelines for improving the functioning of groups, and fostering fair and equal participation and evaluation. In a manner similar to Gates et al she suggests requiring students to prepare a systematic dossier of all their individual work, which they must use as a "ticket" permitting them to participate in group work. She also suggests assigning individual grades based on a combination of the group grade, an assessment of the individual's rough work, a "division of labor" report produced by the group, and peer ratings of the team members.

Layton and Ohland [17] found no gender effects but did find race effects when students rated their peers as part of group projects. In [18], they modified their peer-rating instrument and added additional related instruction. Using this approach, they found no race effects, but they did find gender effects, indicating that their new approach was only partially successful. Sims-Knight et al [19] examined the effect that an assessment-based continuous improvement process can have on team skills when team projects are undertaken in software engineering courses. They found that "self-assessments of both knowledge base and team process plus team training reading and exercises were not sufficient to promote improvement even in basic declarative knowledge."

A theme running through several of the above authors' work is that whatever strategy you use, it must be explained openly to the students and the students should be given a say in assessing the effectiveness of the strategy. The well-known book on college classroom teaching by Angelo and Cross [1] provides some techniques for doing this.

3. Approaches to grading individual effort

This section will discuss grading criteria, grading individual effort within groups, handling team breakdown, and evaluating project-grading methods. Instructors should be able to use this section to derive and evaluate their group grading schemes. The points made in this section have been synthesized from the literature as well as the experiences of educators at our universities.

3.1 Criteria for good grading schemes

Many of the criteria for grading have been touched upon in section 2; the grading scheme should:

- Be fair
- Be consistent
- Reflect achievement of educational objectives
- Provide good, understandable feedback
- Encourage students and avoid unnecessary discouragement
- Not contribute to grade inflation or deflation
- Be easy (as regards workload) on the grader
- Control grading volatility, sensitivity, and risks
- Be accurate and unbiased across a wide range of project types, and finally
- Discourage "risk managing" among students towards grading criteria (e.g. minimizing effort).

Let us examine some of these more closely. First, for the scheme to be fair and consistent means that grades accurately reflect differences in accomplishment from group to group and among members of a group. The grades must take into account appropriate information (particular presentations, reports, exams, etc.) so as not to unreasonably penalize or reward particular students or groups. For example, a grading scheme that rewarded only the results of a "final product demonstration" might unfairly benefit a group in which all the work was done by a single programming guru, especially if the other team members learned little and the resulting system was undocumented and unmaintainable.

There should be no numerical anomalies caused by a grading formula used; that is, an increase in skill or knowledge should map to an increase in grade. Note that anomalies can be caused by some kinds of "step functions" or strong non-linearities. The scheme should be designed so that it cannot be "manipulated" by students in any way to thwart fairness, e.g. colluding with other groups, skipping things that don't count, etc. The scheme should minimize opportunities for cheating and be as objective as possible, so as not to introduce grader bias or arbitrariness.

Designing a scheme to ensure that grades reflect achievement of educational objectives is challenging because sometimes it is hard to avoid rewarding mere volume of work or pure effort (assuming that actual knowledge and skills are the real educational objectives).

Providing good, understandable feedback implies that students and groups know what they have to do, why they obtained their particular grade, and what they can or have to do in the future.

Regarding discouragement, students may be given the opportunity to make up for some bad early results if they do well later. Also, a student who has trouble working in a group through no fault of his/her own should still be able to obtain a fair grade.

At the University of Southern California, we have found that it is both practical and fruitful to utilize our real-project, client-based CS577 course as a software engineering "laboratory" to carry out software engineering experiments and collect research data outside the course educational objectives. The activities range from observational studies such as of development processes and effort distributions [11] and point interventions such as perspective based reading techniques [12], to longitudinal studies and fundamental process changes such as use of COTS [13] or schedule as an independent variable [14]. As a result, the last three items listed above have become essential in ensuring the validity of the experimental data and results in addition to expanded emphasis on the previously listed criteria. The experiments often continue or are repeated from semester to semester. While some of the additional criteria are self evident, some elaboration is in order. There are a multitude of uncontrollable factors within software engineering project courses such as project size and scope, quality of client, organizational politics, technology changes, and so forth. Volatility and risks in these factors are natural in real word projects and thus we do not attempt to control them, which might invalidate our experiments. Our grading must, however, be robust in the face of these uncontrolled factors.

3.2 Techniques for grading individual contribution

There are numerous ways to approach evaluating individual contribution, some of which were touched on in Section 2. Some grading schemes follow:

1) The group mark is everybody's mark.
2) Everybody reports what they personally did, and separate marks are given to those components by the grader. As mentioned, McKinney [5] suggests that this can be done by using either a dossier of rough work or a division of labor report produced by team members.
3) Other group members report (confidentially or openly) the relative contributions of other group members to allow for an adjustment of the final grade.
4) Pop quizzes in class to ensure that students know the intimate details of the project.
5) Cross-validating with the results of individual work (possibly reducing the weight of group work for students who perform poorly on exams or individual assignments).

We will now examine some of these approaches. It appears that 1) is probably the simplest approach, and probably the most common. It certainly is easy on the grader. However, it may be the least fair approach and may discourage students who have a "poor" team. Scheme 2) is more work for the marker and is not normally possible to completely implement. Also, this scheme is somewhat prone to manipulation (unless a dossier is presented, students can protect their peers by lying about what they and others did). For scheme 3), one way to combine the grades would be to have each team member grade the other members of the team on a scale of 100 and the average of these scores could be used to adjust the final grade. In large groups, it might be wise to drop the highest and lowest grades assigned by peers to reduce opportunities for collusion.

We have had interesting experiences with these approaches at our three universities. As a variant of scheme 2), Jurek Jaromczyk of the University of Kentucky (UK) requires each student to prepare and update a web log weekly. He then monitors this web log to track progress. This requires more effort for the grader, but is hard for the student to manipulate.

Paul Piwowarski of UK also required each project to have a web page. Among other things, the page provides the weekly activity of the group and of individuals. This is monitored during the semester. Despite using this scheme, Piwowarski ended up giving the same project grade to all team members: It was the first time he had taught the course and he did not feel that he had enough justification and proof to give disparate grades.

Tony Baxter, also of UK, uses the pop quiz method (Scheme 4). This requires considerable effort on the part of the grader, because he or she must be knowledgeable enough and up to date enough on each team's project to be able to detect a student's ineptitude.

The University of Southern California uses a combination of these approaches. Grading scheme (1) is utilized within the four major milestone reviews (LCO, LCA, RLCA, IOC see [15]) of project artifacts (two per semester). For scheme (2), there is weekly reporting of individual effort and a heavily weighted final individual project critique that asks a comprehensive set of questions. The same individual project critique asks students to discuss management and staffing issues for which low performing team members are commonly accounted for as does the bi-weekly project progress reports. We also ask the students to assess their own individual contribution for which a significant number of points are either applied on behalf of the student or are in part redistributed to deserving teammates. Finally, the various grading results are cross-validated with each other and the individual effort due to homework, project review presentations, and quizzes. A particularly notable pattern we've identified is when a student submits a poorly detailed individual critique, void of any real content or even constructive criticism: of the course, instructors, or teammates. When the answers are "the project was good. I learned a lot," we suspect that such a student was not engaged with their team and likely not to have contributed fully. We look at the entire team's individual critiques which often will indicate a low-performing team member as the cause of project shortfalls. We would then validate further by considering the students individual effort. Generally there will be many missing or low-scoring marks, further validating that this student was not engaged with the project.

At the University of Ottawa, our standard approach in the capstone project [4] is based on scheme 1), with scheme 3) being applied in exceptional circumstances. In a second-year software engineering course requiring compulsory group work [9], we apply approach 5), increasing the weight of exams for any student whose exam grade drops below 66%; exams are weighted at 100% of the mark if the exam mark falls below 50%. One advantage of this is that it requires absolutely no extra work on the part of the faculty member. It reduces animosity against slackers since all group members know that slackers will often do poorly on exams. It is essential, however, that the project be 'standard' so that material learned doing the project can be examined.

The University of Southern California used a hybrid of schemes 1,2,3, and 5 and found that project reviews and individual critiques worked very well. Each project team prepares a structured 80-minute presentation for the "architecture review board" reviewing their project based on the lifecycle anchor points [15]. The rule is that each team member must present something and they are individually graded in regards to their preparation and presentation content. Underperforming team members are readily recognized as they are unable to present appropriate detail on their project and are unable to answer questions effectively. One of the particularly salient benefits of this approach is that it serves as an early warning and intervention process. Through direct feedback and experience of their position in relation to their teammates during the review, underperforming team members often become motivated to improve before falling in to a tailspin from which they cannot recover. As instructors, we can note early on students that may require additional guidance and can help their teams risk manage potential consequences of an underperforming team member. The individual critique serves as a final measure of how engaged a student was in their team and applied material from the course. As noted previously, underperformers are easily spotted as they tend to respond with vagueness and generalities or muddled details. By the time we note this in a student's critique, it is too late for that student to improve. A low score on a critique assures a lower grade in the course than the student's more fully engaged teammates and we use critique results as cross-validation in combination with their project review presentation scores for allocating their individual contribution points to teammates. When confronted with overwhelming evidence from poor scores on review presentations, critique, and (usually) homework, underperformers generally confess their lack of participation and are eager to provide restitution by sacrificing their individual contribution points.

We have found that it does not work well to directly ask students to evaluate their teammates (scheme 3) as students are reluctant to provide details or "rat out" their teammates. This is generally unnecessary, as noted earlier we have multiple other means of spotting underperformers. What we do instead is ask students to evaluate within their critique the impact of problems they encountered, including team personal. Rather than "name names" they are more comfortable and candid with describing hardships such as "we would have been able to implement all the high-priority requirements if all the team members performed to the expected level." This provides critical information needed to gauge how much of an underperforming students individual contribution points should be allocated to the rest of the team – that is, student evaluations of other students are best used in *assessing magnitude* and not *identification*.

The University of Kentucky used scheme 3) and found that it worked very well [16], though this is contrary to the experience at USC. It was used in several variations. In one variation, an evaluation instrument was used. The students evaluated themselves as well as each team member. This instrument was used in three different

Table 1. Grading criteria mapped to individual effort grading.

Grading Criterion	All same grade	Each reports own effort	Each evaluates self and others	Each reports in weekly Web log	Each reports in journal	Quiz in class	Project reviews	Individual effort analysis
Fair			X			X		X
Consistent	X					X		X
Reflect Educ. Objectives				X		X	X	
Provide feedback		X	X			X	X	
Encourage students		X					X	
No grade inflation				X				X
Easy on grader	X	X	X		X			

software engineering project courses with undergraduate students and graduate students. Each semester there was at least one team that felt that one or more team members were not "pulling their weight." In all but one instance, the instructor had to serve as an intermediary. In two incidences, the "idle" team member acknowledged that they were not doing their share and their grade was lowered appropriately. In one incidence, the "idle" team member "vanished" from class and did not complete any further assignments or exams and received an appropriately low grade. In one incidence, the "idle" member "kicked it into high gear" and earned the respect of the other team members by over-performing at the end of the project. This pattern is not what we should promote as instructors, but at least solves the unfairness problem.

The above scheme seemed to work well in that students were not reluctant to comment on an "idle" member. However, the instructor worried that the scheme may also have encouraged team members to "gang up" on another member. The scheme also did not allow frequent monitoring of the teams as the forms were only turned in three times throughout the semester. This scheme might be improved by combining it with other schemes and variations, such as having the students e-mail such evaluations to the instructor weekly. This would require more work on the part of the instructor, though.

The instructor found that one aspect of the project assignment was very good at uncovering students who had not contributed much: the project demonstration. The instructor required all team members to participate in this demonstration in the instructor's office or student lab. The instructor had developed an "acceptance plan" and asked the team members to perform the tests and/or the instructor performed some tests. The instructor also asked each member questions about the project and asked for demonstration of certain features. The demonstrations often "rooted out" students who did not know much about their projects (at the very end of the project schedule, no less).

Variations on this scheme such as weekly web logs, periodic journals, and random quizzes were also used at UK. It was found that the weekly web log and quizzes worked well. The web log needs to be more formal though. In the future, UK plans to tell all group members to review the web log for accuracy (of all members' accounts) as it will affect their individual grades on the project. The quizzes also need to contain more detailed, team-specific project questions.

In Table 1, each of the above grading schemes has been mapped to the grading criteria discussed earlier. Note that no single scheme meets all the grading criteria. It seems clear that using a combination of these schemes is the best approach for achieving adequate criteria coverage. The grading schemes may be combined in a number of ways:

- A fixed percentage of each person's grade is based on scheme 2) or 3); the remainder on scheme 1), e.g. S1*0.9 + S3*0.1.
- The group grade, scheme 1), is "multiplied" by a factor from scheme 3), e.g. simply S1 * S3/100.
- A fixed number of grades are available to be distributed among all the group members. The members decide who gets what. For example, the instructor grades a project (using scheme 1) as 90%. For a 3-person group this means that there are 90 * 3 = 270 marks that the 3 students can distribute among themselves in whatever manner they agree on.

Whatever the grading scheme, we reiterate the following issue raised by other researchers in section 2.1: It is important that the syllabus and/or grading policy clearly spell out the instructors' intent. The University of Kentucky often uses this paragraph:

"Group Projects:

The group project for the course will require you to work together with other students in the class. You will be evaluated on your contribution to the group project and presentations of the project results. The instructor will make group assignments. Group members are not guaranteed to receive the same grade; evaluation of the group will be individualized to determine individual understanding, commitment, and mastery of the project goals. As part of the project, written reports will be required. Proper language usage is required."

3.3 Preparing for when bad things happen

Generally there are some warning signs when things are not going well within a team. One or more disgruntled students will call, e-mail, or visit an instructor. At this time, we believe that the instructor should encourage the students to work it out among themselves. If this fails, the instructor can serve as an intermediary and meet with the team.

Hong-Mei Chen at the University of Hawaii [8] approaches the problem very much like industry. Once a student is part of a team and is not pulling his/her weight, the team can "fire" them. The “fired” student can then try to get rehired by another team. If unsuccessful, the student will be given an individual project to work on. If the latter occurs, it becomes labor intensive for the instructor and grader. A variation on this is that fired members could be assigned a 0 or a failing grade.

4. Suggested best practices

Based on the findings above, we feel that there are several best practices that can be suggested.

- Allow team members to evaluate each other, but carefully and/or frequently monitor this to prevent the “mob” mentality or collusion.
- Use project demonstrations and/or quizzes to further test project knowledge.
- Require frequent generation and/or posting of individual effort information, or the maintenance of individual dossiers
- Cross-validate individual effort evaluations with group evaluations
- Have multiple individual grading methods (e.g. individual project critique and individual effort) to enable adjustments for individual contribution
- Whatever grading scheme you use, evaluate it using the criteria we listed in section 3.1

5. Summary and conclusions

In this paper we have discussed how software engineering course instructors can tackle problems associated with grading group projects. We first pointed out that group projects are an essential component of software engineering curricula: Students must practice collaborating with teammates since this is what they will be doing in their careers.

Whenever an instructor is designing a grading scheme, he or she should evaluate the scheme according to the criteria we presented in Section 3.1. Among these criteria are issues of fairness, consistency, encouraging students, and preventing various kinds of anomalies. The fairness issue is one of the most important since studies have shown that students feel bitterness about slackers.

In Section 3.2, we examined some techniques and grading schemes that can help instructors meet the criteria. The various mechanisms encourage accountability or permit individual contributions and knowledge to be properly assessed. We suggest combining several techniques.

6. References

[1] Angelo, T.A. and Cross, K.P., “Group Work Evaluations”, in *Classroom assessment techniques: a handbook for college teachers*, Jossey-Bass Publishers, a Wiley company, 1993, pp. 349-351.

[2] Gates, A.Q.; Delgado, N.; Mondragon, O., “A structured approach for managing a practical software engineering course”, 30th Frontiers in Education conf, IEEE, 2000, pp. T1C/21 - T1C/26.

[3] Johnson, D. W., Johnson, R. T., and Smith, K. A. “Cooperative learning: Increasing college faculty instructional productivity”, *ASHE-ERIC Higher Education Report* 20, 4, Graduate School of Education and Human Development, The George Washington University, 1991.

[4] Lethbridge, T. “SEG 4000: Rules for Projects”, University of Ottawa, web page as of Sept. 2002, http://www.site.uottawa.ca/~tcl/seg4000/rules.html

[5] McKinney, K. “Tips for Grading Group Work”, Illinois State University, web page as of Sept. 2002, www.cat.ilstu.edu/teaching_tips/handouts/ tipsgroupwork.shtml

[6] Schultz, T.W., “Students assessing teams”, proc. 29th Frontiers in Education conf, IEEE, 1999, pp. 13B2/1 -13B2/3.

[7] Speck, B.W., “Pedagogical Support for Classroom Collaborative Writing Assignments”, *ASHE-ERIC Higher Education Report*, 28, 6, Jossey-Bass, a Wiley Company, 2002, pp. 1-139

[8] Chen, H-M., web page http://www.cba.hawaii.edu /HMCHEN/home.htm

[9] Lethbridge, T.C., University of Ottawa, SEG 2100 course web page: http://www.lloseng.com/seg2100

[10] Individual Critique Guidelines for CS577a 2000 http://sunset.usc.edu/classes/cs577a_2002/IndividualCritiqe.html

[11] Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., and Madachy, R., “Using the WinWin Spiral Model: A Case Study,” Computer, July 1998, pp. 33-44.

[12] Shull, F.; Rus, I.; and Basili, V.R. “How Perspective-Based Reading Can Improve Requirements Inspections”. IEEE Computer 33, 7 (July 2000), 73-79.

[13] Port, D., Bhuta, J., Yang, Y., Boehm, B., “Not All CBS Are Created Equally: COTS Intensive Project Types,” Accepted to ICCBSS 2003.

[14] Boehm, B., Brown, W., “Mastering Rapid Delivery and Change with the SAIV Process Model”, Proceedings, ESCOM 2001, April 2001

[15] Boehm, B. (1996), “Anchoring the Software Process,” IEEE Software, July, pp. 73-82.

[16] Hayes, J. Huffman. “Energizing Software Engineering Education through Real-World Projects as Experimental Studies,” in *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET)*, Covington, KY, February 2002.

[17] Ohland, M. W., Layton, R A.. “Comparing the Reliability of Two Peer Evaluation Instruments”. Proceedings. ASEE Annual Conference & Exposition, St. Louis, MO, Jun. 2000.

[18] Layton, R. A., Ohland, M W., “Peer Ratings Revisited: Focus on Teamwork, Not Ability”. In Proceedings. ASEE Annual Conference & Exposition, Charlotte, NC, June. 2001.
[19] Sims-Knight, J., Upchurch, R., Powers, T.A., Haden, S., and Topciu, R. “Teams in Software Engineering Education”. Proceedings of Frontiers in Education 2002, November 6 – 9, 2002, Boston, MA, pp. S3G17 – S3G22.

7. Acknowledgments

We would like to thank Hong-Mei Chen of the University of Hawaii, Jurek Jaromczyk, Paul Piwowarski, and Tony Baxter of the University of Kentucky for their insights and contributions to this paper.