Fairly Assessing Team Members' Contributions

By: Team Uniqua

A common issue in university and corporate software development teams is the lack of concrete methods to evaluate group members. No measurable values, such as hours spent or lines of code submitted, can truly capture the contributions of team members accurately[(1)]. In general, the only ones that know how the work is divided are the ones working in the group. We provide a solution called the 100/N method. This is a viable solution used to evaluate individuals based on the end product's net worth adjusted by a coefficient determined by those involved. The only constraint is the sum of all evaluations equal the total project worth.

Logically, a developer's work can be worth at most the end product, which occurs if he contributes 100% to the development and no one else contributes. This places a monetary value upon individual developers for their work over a period of time in which companies can use to evaluate and adjust decisions such as salaries and promotions. As a concrete example, for a $50,000 project, one developer receiving an average group evaluation for contributing to 50% of the work will imply his work over the time period has been worth $25,000. On the other hand one with an evaluation of 5% will only be worth $2,500. Corporate choices such as promoting and laying off can be determined by this statistic. Classroom settings are similar but slightly less trivial mathematically as it is subject to additional considerations such as failing scores and maximum scores. For instance, a project for 4 people is completed 50% by one individual, but the maximum grade we can allocate is only 100%. Thus we will hereby explore group assessment in terms of grades. The underlying principles are the same though the mechanics for the math is less stringent.

Our primary approach to the method of evaluation is to allow each team member to allocate 100 credits to group members including themselves as the scale factor for the grade each receives. Take a 4-member team for example. If everyone have contributed the same amount of work, each person will allocate 100/4=25 credits to each member (including themselves), thus everyone ends up with 100 credits (25 from each person) and will receive 100% of the group grade. Stated in plain English, everyone in the group contributed equally. Suppose one member did barely anything. The group members each give him 10 credits, and then split the remaining evenly among others (30 each). The one free-loader allocates 25 credits to each. Then what results is the free-loader will be scaled with a final grade of [(3*10)+25] or 55% of group grade, while the rest receive 115% of group grade. This implies 3 members worked hard and their overall effort was worth 115% of the group grade, while the free-loader was clearly unproductive and unhelpful. However, this method poses some problems in which modification is needed.

The first modification here is we realize any member can allocate 100 credits to himself and receive a grade at least as high as the group grade. For less extreme cases we realize anyone can benefit by deviating from the true evaluation. The solution to this is to not count each person's own personal evaluation and rescale remaining credits to add up to 100. In the above example, suppose the free-loader gives himself 40 credits and 20 each to the others. Then his credits are discarded and the remaining are re-scaled to 33 each. For the other group members, their own credits are discarded (30), and the remaining {30, 30, 10} is rescaled to {43, 43, 14} to be out of 100. The free-loader than receives a scale-factor of 42% and the rest 119%. In the case that everyone contributes an equal amount, this method results in everyone receiving the group grade. Notice that regardless of what each person gives themselves, what matters is the ratio of work they believed the others contributed with respect to each other.

For corporate development teams, the scale factor determines contribution rate in the team. Multiplying the scale factor by the average project net worth per person—defined here as P/N where N is number of developers and P is net-worth of project—determines the value of a developer's work. It is not completely necessary though, as the scale factor itself usually produces a decent comparative statistic for evaluating contributions.

After our first major modification, this evaluation method has ruled out opportunities to unfairly impact one's own grade. The method also rules out possibilities for group collaboration for improving everyone's grade. In the usual peer-evaluation system where arbitrary numbers 1-5 are used to symbolize different degrees of contribution), groups tend to talk to each other and try to raise everyone's averages(1)(2. In the current method, giving someone a higher score implies lowering another's. Thus the only logical solution is to provide an evaluation true to their beliefs.

However, while it is true only the developer themselves know how the work was divided, students (unlike professionals) tend to have faulty beliefs, which results in a larger deviation when estimating the difficulty, workload, and quality of other people's work(3)(4). Even in a corporate environment, most programmers tend to have different and specific skill sets that make evaluation of others less credible. In order to counter this problem, it is important to include a strict criteria in the way the 100 credits should be handed out. It is important to have a criterion especially designed for the project, suggesting possible difficulty and workload of tasks. Credits can also be suggested to be given to people who gave valuable feedback when questions are posed and participated actively in discussions. The existence of a criteria relative to the project can encourage more accurate considerations during evaluation and deviate less from judgments given by a professional(3).

While anonymity seems to encourage honesty in reports, studies have shown relatively no effect on peer evaluations(3). It appears that open evaluations also happened to encourage honesty as individuals perceive the need to defend their point of view in the future. However, one critical process during evaluation is that they are done better in a monitored environment where no one has access to the results of other evaluation forms. It appears that by having access to other results, self-defensive behavior may result and produce results noticeably more biased(3). This bias is not restricted to self-serving bias. Most of the time this bias tends to deviate from the result of a credible estimate.

The method for evaluating contribution presented so far will work for any class project and in most Agile development groups. It focuses on unbiased evaluation methods based on technical knowledge of the software in development, and eliminates the possibility for unfair collaboration. However, two problems remain. Firstly, we need to realize the scope of this evaluation method lies tightly within the boundaries of each individual development team. Because evaluations are done based on knowledge of other people's work, this cannot work for multi-departmental projects or large scale projects involving many teams. Secondly, no one is set to decide whether or not the final evaluations are correct. Should there be a case where everyone overrated or underrated one individual's work, this will show up in the evaluation. The solution for both these problems is to include team/project managers(1)(5).

The main role for each manager is to lead and oversee the project. Similar to any manager position, they need to be qualified individuals experienced in the field. They will be able to have accurate and, more importantly, credible estimates for evaluations. To emphasize their importance, managers should have their evaluations be weighted greater than the individual developers. Their daily roles would include keeping track of everyone's tasks, accomplishments, and assigning jobs. The immediate result of including this person is a credible basis for checking the validity of the evaluations. The criterion in which was mentioned before will be created by the manager. He will be able to hint at specific areas where higher credits should be awarded and point at distinguished/outstanding contributions in the group. In large-scale development, there will be many project/team leaders. Together they are a team, and they will have a leader/manager of their own. Their method of evaluation would be identical to their team, except now they rest only as a team member. This hierarchy can technically grow indefinitely.

The methods we introduced here can be used outside the context of software development. It is most suitable for evaluating group members where there is a lack of concrete data. Finance employees can be rated based on the money they make for the company. But for software development in or outside of class tends to be more fuzzy in a group setting. The 100/N credit system builds on top of previous peer-evaluation systems, eliminating features that cause unfair and biased results. Applicable to both University and Corporate settings, this guarantees fairness and accuracy for evaluating contributions of team members.

Bibliography

(1) Garner, William B., “Assessing Individual Contributions to Group Software Projects”, University of Guelph, <http://www.cs.ubc.ca/wccce/Program03/papers/Gardner-Group/Gardner-Group.htm>

(2) Ruble, Thomas L., Sigfredo A. Hernandez, William J. Amadio, “A Comparison of Peer Evaluation Systems in Team-Based Learning”, 2004, Rider University <http://abe.villanova.edu/proc2004/ruble.pdf>

(3) Omelicheva, Mariya Y., “Self and Peer Evaluation in Undergraduate Education: Are Promises Worth Risking the Perils?”, ASPA Inaugural Conference on Teaching and Learning, Washington, DC, February 2004

(4) Demerjian, Marlene D., “Determining Faculty Grading Variation”, College of the Canyons, Santa Clarita, Ca, June 1995 <http://eric.ed.gov/ERICDocs/data/ericdocs2sql/content_storage_01/0000019b/80/14/22/7b.pdf>

(5) Bogue, Robert, “Anatomy of a Software Development Role: Project Manager”, <http://www.developer.com/mgmt/article.php/11085_3526491_2>