Yang, J. C. et al. (Eds.) (2018). Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education

Git as a support to assess students' contribution in teamwork

Mélissa Clarisse NTIRANDEKURA[a] & Thierry EUDE[a*]
[a]*Dept. of Computer Science and Software Engineering, Laval University, Canada*
*Thierry.eude@ift.ulaval.ca

Abstract: Teamwork is one of the ways to develop the transversal competencies expected in industry, especially in the field of Information Technology. However, the success of teamwork during programming courses depends on the choice of tools that facilitates collaborative work, organization and conflict management. Objectively evaluating both the individual and the collaborative work must be a great part of this choice. Among these tools, we propose to retain the version control system Git. Indeed, Git facilitates teamwork evaluation by giving access to information recorded in its history. It then becomes necessary to define the criteria to be applied to make an analysis and a judgment. We propose in this article an inventory of the potential evaluation criteria and identify the most relevant ones in terms of evaluation of the regularity and quantitative contribution considering the biases that they induce. Moreover, one of the difficulties that teachers may encounter when producing an assessment is not only to evaluate contributions, but also to be able to monitor them regularly, especially when teams are numerous. We then recommend a support tool based on the criteria we identified, offering a general overview and facilitating access to details. Its evaluation as part of a course allowed us to analyze different team profiles as well as the limits to the use of such a tool to make a judgment.

Keywords: Transversal competencies, computer science education, Git, evaluation criteria, teamwork assessment

1. Introduction

During academic training, teamwork is one of the best ways to acquire and consolidate transversal competencies, especially in the field of Information Technology. However, the success of teamwork during programming courses depends, among other things, on the choice of tools to facilitate collaborative work, the organization of teammates, their collaboration and their management of conflicts (Brun, Holmes, Ernst, & Notkin, 2013). The possibility of being able to objectively evaluate both the individual and the collaborative work of the team members must be a great part of this choice (Laadan, Nieh, & Viennot, 2010) (Haaranen & Lehtinen, 2015). Lanubile et al. (Lanubile, Ebert, Prikladnicki, & Vizcaíno, 2010) have proposed different tools in this direction. Among these tools we will retain the Git version control system. The next step is to identify the criteria that can be considered to quantify contributions during teamwork. By having a support tool that allows for quick monitoring of each team, a teacher will be better able to identify a team that is going through a period of conflict. This then requires an analysis, and an agreement on the right decision to make.

2. Evaluation Criteria

In order to be able to evaluate the regularity of the team, based on the Git repositories, the use of a visual illustration or the result of specific git commands can be used to observe the degree of absolute or relative contribution of each member over time in terms of lines of code or commits. It is important to be able to know the detailed contributions of each member in terms of volume that can be associated with a quantity of work. The number of commits can indirectly represent the regularity by measuring their frequency, but also the contribution by measuring their number. The number of files can be a good indicator of individual contribution but it should be associated with other criteria

such as the number of commits. The number of lines of code can be a good indicator of contribution. However, not all lines in a source code match the same degree of contribution. Also, there are several variants of SLOC (Source Line Of Code) (Bhatt, Tarey, Patel, Mits, & Ujjain, 2012) but only some of them can be considered as a contribution measure. It therefore appears that, in practice, only the actual lines of code, in other words, the executable physical can be used on the basis of the information collected from the Git repositories as a criterion for evaluating the contribution of each team member. In addition, these lines of code completed by the commits are the best criteria to retain when evaluating teamwork.

3. Gitanalysis : a Tool to Support Assessment

One of the difficulties that teachers may encounter when producing an individual assessment of the knowledge and skills actually developed by their students during team-based programming work is not only to be able to evaluate their respective contributions, but also to be able to track them regularly. Having a support tool that offers a general overview and facilitates access to details becomes essential, especially if the teams are numerous. Gitanalysis is a tool allowing a global vision of the regularity and the quantitative contribution of all the teams in the form of a scoreboard which can represent all the teams and their members as shown in **Error! Reference source not found.**.

Teams		Activity indicator	Last activity	Regularity	LOC added		LOC deleted		Commits	LOC per commit
Loutre		↑			1137	100%	301	100%	35	
	[illegible]in	↑	4	1	218	19.17%	77	25.58%	11	26
	[illegible]nas	↑	5	4	219	19.26%	78	25.91%	4	74
	[illegible]on	↑	1	1	441	38.79%	80	26.58%	11	47
	[illegible]esA	↑	1	1	259	22.78%	66	21.93%	9	36
Studio		→			13600	100%	4993	100%	110	
	[illegible]on	↑	0	4	7329	53.89%	2916	58.4%	53	193
	[illegible]rier	→	12	1	4064	29.88%	1088	21.79%	21	245
	[illegible]own	→	13	3	348	2.56%	69	1.38%	10	41
	[illegible]nal	↑	0	4	1859	13.67%	920	18.43%	26	106
Gaff		↓			5663	100%	2889	100%	90	
	[illegible]ais	↑	1	6	2021	35.69%	1255	43.44%	28	117
	[illegible]let	↑	1	1	3251	57.41%	1395	48.29%	56	82
	[illegible]a912	↓	62	4	391	6.9%	239	8.27%	6	105

Figure 1. Evaluation of teams by Gitanalysis for a period of 4 months on java files (extract)

In order to be free from opportunistic activities, the regularity is determined as the average of the differences between the last 4 days when commits were made. Each member can then be classified according to his regularity according to intervals in number of days of activity. Thus, the member is considered regular, less regular or irregular if the regularity is found respectively in the intervals of [0, 7], [8, 10] or [11, ∝[. Indeed, if the member spends more than 10 days without making new contributions, he will potentially encounter difficulties in closing the gap.

The last activity is evaluated as the number of days elapsed between the date of the last commit and the date of analysis which is by default the current date or the date the user selects to determine the last activity of each member of the team. It is considered recent if it is in the interval of [0, 7], contemporary if it is in the interval of [8, 10] and old if it is in the interval of [11, ∝ [.

The activity indicator is determined from the last activity and the regularity. It allows classifying a member as an active member, a member to supervise or a member at risk.

From the quantitative contribution made by each member of the team, Gitanalysis also determines that of the entire team. Indeed, we have the total number of LOC (Lines Of Code) added and deleted by all team members as well as the total number of commits. Gitanalysis is also based on the activity indicator of each member to evaluate that of the entire team. We can then distinguish functional teams when most of members are active ("Loutre" team in Figure 1), temporarily dysfunctional teams when most of members are to be supervised ("Studio" team in Figure 1) and dysfunctional teams when at least one of the team members is at risk ("Gaff" team in Figure 1).

4. Limits

First, when analyzing the contribution of each member of the team, taking into account some lines of code added can skew the result of the analysis. Indeed, if a member adds a library, it will be counted as a contribution. Second, if a team member uses multiple computers to push his code and his computers are configured with different names and emails, Gitanalysis will not consider it is the same person who shared his work. So, to avoid this case, teams would be advised to configure Git with the same name and email on all the computers they would use. Third, to be well evaluated, team members should make regular commits each week, to prevent their teams from being evaluated as being temporarily dysfunctional or dysfunctional teams. Fourth, considering the modified lines of code as a great contribution to teamwork, when handing over the work, is not necessarily true because the member who started by adding lines of code is often the biggest contributor. Finally, dysfunctional teams are to be considered with caution. If one of the members drops the course, it penalizes the rest of the team. Thus, for a dysfunctional team, when at least one of the team members is at risk, it would be necessary to decide if it represents an abandonment and recalculate the activity of the team by omitting the abandonment.

5. Conclusion

In computer science education, teamwork is a means of developing the transversal competencies expected in the industry. However, the evaluation of this work as part of a course remains a real challenge for teachers, especially if the teams are numerous. We propose in this article the use of the Git version control system as support. After reviewing the possible criteria, it appears that the commits and LOC are the most relevant ones when assessing the team's regularity and the quantitative assessment of the contribution of each team member. We then proposed a typical application, presenting a scoreboard of contributions. This support tool provides a quick overview for the teacher who uses it to keep track of team progress, while highlighting members in difficulty. It then facilitates monitoring.

Also, for the teacher to make a final judgment in order to establish a summative evaluation, this tool should be used in conjunction with peer assessment.

Acknowledgements

T. Eude gratefully acknowledges the financial support of Cisco.

References

Bhatt, K., Tarey, V., Patel, P., Mits, K. B., & Ujjain, D. (2012). Analysis of source lines of code (SLOC) metric. *International Journal of Emerging Technology and Advanced Engineering, 2*, 150-154.

Brun, Y., Holmes, R., Ernst, M. D., & Notkin, D. (2013, 10). Early Detection of Collaboration Conflicts and Risks. *IEEE Transactions on Software Engineering, 39*, 1358-1375.

Haaranen, L., & Lehtinen, T. (2015). Teaching Git on the Side: Version Control System As a Course Platform. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 87-92). ACM.

Laadan, O., Nieh, J., & Viennot, N. (2010). Teaching Operating Systems Using Virtual Appliances and Distributed Version Control. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 480-484). ACM.

Lanubile, F., Ebert, C., Prikladnicki, R., & Vizcaíno, A. (2010). Collaboration tools for global software engineering. *IEEE software, 27*, 52.