Version Control System Gamification: A Proposal to Encourage the Engagement of Developers to Collaborate in Software Projects

Alexandre Altair de Melo, Mauro Hinz, Glaucio Scheibel,
Carla Diacui Medeiros Berkenbrock, Isabela Gasparini, and Fabiano Baldo

Santa Catarina State University, Brazil (UDESC)
Computer Science Department
Graduate Program in Applied Computing
Joinville, SC – Brazil
{alexandremelo.br,mmhinz,gscheibel}@gmail.com,
{diacui,isabela,baldo}@joinville.udesc.br

Abstract. This paper proposes to use gamification for recognition of software developers' collaboration and commitment. In order to improve productivity, the paper also evaluates the users' engagement in a software development project. The idea is to use the information extracted from source repositories where developers realize their commits. A tool proposes ranking via news feed that will extract information from the source repository by using software engineering metrics, such as McCabe's cyclomatic complexity, in order to build a ranking system, which highlights and rewards the most active developers. The ultimate goal is to determine whether the use of gamification encourages collaboration and commitment of all involved in software development projects.

keywords: gamification, collaboration, version control, software engineering, interface.

1 Introduction

Over the past years, the term gamification has been calling attention to be applied beyond gaming environments and carry the approach to other collaborative tasks, such as software development. The classic concept of a game can be used to verify if developers and others involved in the projects promote engagement, reach their goals and even collaborate, resulting in increased productivity. The term gamification was coined by Nick Pelling in 2002 [12] and is conceptualized as the use of design techniques, thoughts and game elements to improve the experience in non-gaming situations.

Gamification is the application of game metaphors in not-ludic contexts to influence behavior and increase motivation and engagement. According to Zichermann *et al* [23], gamification is 75% psychology and 25% technology related, and bears a strong incentive for greater engagement of developers. The process of using gaming techniques, when used properly, has the power to engage users by stimulating communication and learning [15].

Companies in the software industry, such as SAP, began applying gamification internally with their development teams, as well as with external partners. However, this approach is also used in other types of applications and solutions, such as Linkedin, opinions from customers on Amazon, Nike+ and Foursquare, which all use gamification to motivate users to perform tasks that require collaboration and providing feedback [8].

If used in software development, feedback can be extracted directly from a version control database tool and the most active developers get rewarded [9]. The use of software allowing the users to connect, interact and be aware of what their colleagues are doing has been successful in building systems and applications, as approached by Treude and Storey [20].

This paper proposes an extension to the work of Singer and Schneider [18], in order to calculate and rank participants involved in software development projects.

This paper is organized as follows: Section [2] presents related works; Section [3] presents key points about the basic mechanisms of games and the concept behind McCabe's metric; Section [5] presents the role of a version control system for the proposed model; Section [6] introduces the proposed model which is an equation that uses the McCabe's cyclomatic complexity to reward developers who modify sources of higher complexity; Section [7] illustrates our prototype to support the proposed model; and Section [8] concludes this paper and presents future works.

2 Related Works

The concept of gamification was initially applied by Website managers as a tool that maximizes customer engagement. Among these initiatives are the examples of Yahoo Answers and StackOverflow. In the first example, anyone can respond to various questions be ranked and rewarded for that. The second one focuses on developers' interaction, rewarding and ranking users according to their responses. Due to the effectiveness of using gamification in non-academic environments, this concept has begun to be studied by scholars [6].

In another study about gamification applied to software development, by Ahmad and Jaafar [1], the behavior of users that apply gamification is analyzed and guidelines for developing gamified applications are proposed in order to assist the area of human-computer interaction. The result of this experiment led to a more participatory experience among users of the study.

In Singer and Schneider's study [18], a preliminary result of an experiment, with encouraged computer science students to make more frequent updates in version control sources, shows that by using a web-based, social networking application software, a newsfeed of commits is provided in a ranking format. Some students were motivated to participate with their bug fixes to earn scores for better ranking; however, other students made superficial commits and it was necessary that the assessment be made by an engineer to check the quality of the commits, which hasn't been completed [18].

The above-mentioned initiatives are recent studies of gamification, since the topic is still new; however, the literature in this area is already well established. When applying the concept of gamification in an activity that does not have the characteristic of being playful, it is necessary to emphasize that the concept of playing is present; therefore, the authors Crawford [4], Deterding *et al* [5] and Salen [17] discussed some standards for the design and development of games. Within these patterns, some features leading to successful games are remarkable and involve challenges, interactions, rankings, points, even social approaches and collaborative actions. The social aspect of games has been well studied in the field of health care and well-being. For example, Velazquez *et al* [21] shows social interaction promoted by games to help the elderly adhere to healthier habits.

Regarding the use of gamification for building collaborative software, Singer and Schneider [18] quote that using metrics based only on commits is simplistic and worthless because it doesn't take into consideration the effort spent on the tasks. Therefore, this work intends to extend Singer and Schneider's proposal for the use of gamification and ranking for building software, through a different form of measurement, which is not only based on the number of commits from developers, but also on lines of source code according to the complexity of its code.

3 Game Engines

Turning an everyday activity into a gaming experience, while making it challenging and fun at the same time, requires a lot of effort. There are several approaches describing the gaming mechanisms. According to Cook [3], a game's mechanisms are based on a system of rules and simulations, which facilitate and encourage the user to explore and learn within many possibilities through the use of feedback mechanisms.

According to Marczewski [12], many real-world tasks can be broken into smaller tasks. In some cases, these tasks can be converted into real games. Koster [10] also comments that games are like puzzles to be solved, like everything else in life, for example, like learning how to drive a car, playing a mandolin or multiplying seven times seven. By adding rules, challenges, opponents and rewards, these activities can turn a task into a much more pleasant and enjoyable experience. In the following sections, we will list concepts inherent in games that can be used to apply gamification in real activities.

3.1 Rewards

According to Groh [7], rewards are used to elevate the reputation of a player within the gaming community, receive a level that releases exclusive access to collected badges and are highlighted in the community. In this process, gamification expands collaboration through competition and boosts the degree of motivation and interest.

3.2 Leaderboard

Leaderboards allow users to monitor their performance in relation to others and can be divided into several subcategories, such as: Global, Friends, Relative, Remote, etc. As pointed out by Deterding *et al* [5], leaderboards represent one of the most common techniques used to create gamification.

3.3 Rules

Rules must be used by players to understand the requirements for reaching each achievement. The set of rules must be clear and acknowledged by the players; however, it must be reminded that a change of rules impacts the players negatively. Therefore, it is important to combine targets and well-define rules.

According to Marczewski [12], a game's rules are vital. When a game is being constructed, a framework and set of rules must be defined. For example, a certain action is worth one point, but a more complex action is worth five points; when reaching one hundred points, the player receives a badge. The goals of a determined game only become clear once the rules are established.

3.4 Feedback

According to Marczewski [12], obtaining feedback is vital in any gamification system. This helps users to check their progress and, periodically, what other participants are doing within the same context. Some forms of feedback are more noticeable than others. Progress bars, points and badges, for example, are some of the fastest ways to provide the status to a user.

4 Metrics

The simplest metric to evaluate the evolution of a computer program is the number of lines of source code produced in it. However, this metric is only useful when used in conjunction with another one, such as measuring the number of errors (bugs) per line of the source code [16]. Furthermore, according to O'Grady [14], it is not successful to measure productivity by using as the only basis the number of lines of codes.

McCabe [13] establishes a framework for measuring the complexity of the source code. This metric is called cyclomatic complexity. Like the idea of Rosenberg [16], this paper considers that the number of code lines changed or added is only relevant when combined with other metrics. Thus, the number of code lines is scored based on the McCabe's cyclomatic complexity of the developed code.

Watson *et al* [22] also shows that high complexity leads to code that is more difficult to test; therefore, it is important to consider that the higher the complexity is, the lower the quality that the software may have.

4.1 McCabe's Cyclomatic Complexity

McCabe's metric is calculated by [13]:

$$M = E - N + X \tag{1}$$

where M is the metric of McCabe's cyclomatic complexity (MCC), E is the number of edges in the graph of the program, N is the number of nodes or decision points in the program's graph and X is the number of the program's outputs.

In terms of programming, the code edges that are executed as a result of a decision are the decision points. Outputs are the explicit return statements in a program. Typically, there is an explicit exchange of functions without explicit return to subroutines.

A simpler method of calculating the MCC is shown in the equation below. If D is the number of decision points in the program, then

$$M = D + 1 \tag{2}$$

In this case, decision points can be conditional statements. Each decision point typically has two possible paths.

5 Version Control System

Version control systems (VCS) are used during the development of most information system projects. While these tools have different features, the main idea is the same; in other words, the VCS are all able to track and store changes to the software from the beginning of its development. Once the software source code is organized into files and folders, VCS can store the changes of the files [19].

According to a survey conducted by Koc *et al* [9], in a version control system, each change is associated with answers to the following questions: who, when and why. As shown in this paper, when we combine VCS with a system of gamification, new questions could arise: Was the developer's contribution relevant to the project? How much has he or she contributed in a period of time? Last, but not least, how can he or she be recognized in the group, in an explicit way because it is clear that his or her contributions and collaborations are relevant?

6 Proposed Model of Gamification

As mentioned in the Metrics section of this paper, the number of modified code lines is only relevant when combined with another metric. In the proposed score model, the number of code lines will be scored according to the cyclomatic complexity of the committed code. For example, using a hundred lines of code added to a simple program have less value than the same hundred lines added to a program of high complexity. Thus, developers who work in critical points of a project receive more points and, therefore, more rewards than developers who modify simple programs.

Following the McCabe's cyclomatic complexity metric, it is possible to evaluate the relationship between complexity and the amount of lines of code modified by a particular developer.

1. The developer commits his modification;
2. A SVN hook is triggered to verify how many lines are modified;
3. The total complexity is evaluated by the hook;
4. The proposed equation is applied and the score is added to the developer's score.

$$Score = (MLoC \times TCC) + ((BCC - TCC) \times (MLoC \times 2)) \tag{3}$$

where MLoC are the modified lines of code made by the developer, TCC is the program's total cyclomatic complexity and BCC is the program's cyclomatic complexity before the developer's modifications.

As mentioned above, measuring the amount of modified lines is only relevant when combined with another metric. The proposal to combine it with cyclomatic complexity differentiates the value of a change in source code, which also follows the Pareto's principle, where 20% of the source code has 80% of the system's total complexity. Joining the amount of modified lines of code with complexity balances the score between simple and complex codes [11], [2].

A commit must be rated according to the overall complexity and the value of the contributions of changes in a more complex source code, not only by the number of committed lines and number of commits made by the developer.

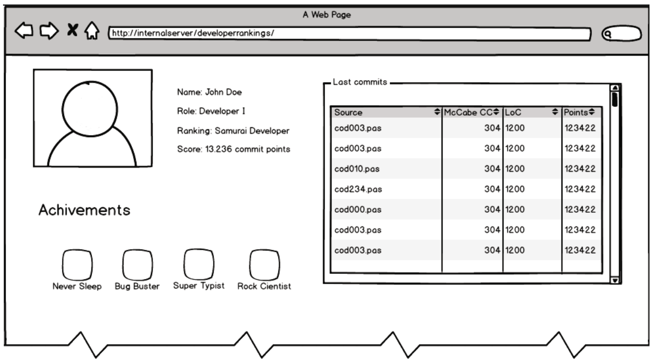
Challenges arise with high complexities. It is known that a high complexity of source code negatively impacts the maintenance process and can induce defects by considerably making more difficulty the executions test; therefore, it is not desirable that developers receive a higher score when using increased complexity, but the score must also be influenced by changes/improvements in complexity.

The equation can be split in two parts: the score by the number of modified code lines, depending on the complexity, plus a positive or negative bonus for improving or worsening the overall complexity. This method helps avoiding that developers are benefited by a high score when increasing complexity; the positive or negative change in complexity will inversely affect the received score.

For example, if a developer modifies 10 lines of code in a source code with a complexity of 15 units, he or she will receive 150 points, but if the previous complexity was 13 units, he or she will lose 40 points for increasing the complexity. However, if the previous complexity was 17 units, he or she will earn 40 points for having reduced the complexity and thus improved the code's quality of testability.

7 Prototype of the Gamification Tool

Figure 1 shows the screen with details of the developer's individual evolution for follow-up, including a small developer profile and the list of source codes that have been modified in conjunction with scores received by such modifications.

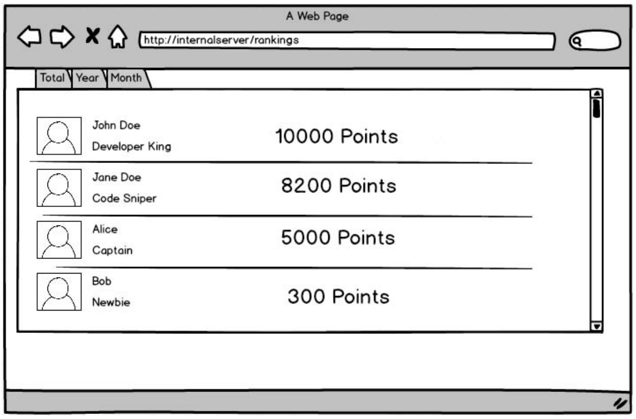


Name: John Doe
Role: Developer I
Ranking: Samurai Developer
Score: 13.236 commit points

Source	McCabe CC	LoC	Points
cod003.pas	304	1200	123422
cod003.pas	304	1200	123422
cod010.pas	304	1200	123422
cod234.pas	304	1200	123422
cod000.pas	304	1200	123422
cod003.pas	304	1200	123422
cod003.pas	304	1200	123422

Fig. 1. Developer's profile

Figure 2 shows the screen with details of the developer group's evolution, according to each one's classification (ranking) and their total points. In this screen, developers can also compare their performances at predefined intervals. Thanks to this comparison, developers gain a sense of competition and will seek to expand their collaboration in the project to reach a higher score.



John Doe
Developer King
10000 Points

Jane Doe
Code Sniper
8200 Points

Alice
Captain
5000 Points

Bob
Newbie
300 Points

Fig. 2. Developer's Ranking

With gamification of the commit process, a plug-in (VCS hook) of the version control system calculates the complexity of source code and extracts the number of contributed or modified lines of code. The number of modifications multiplied by the value of cyclomatic complexity of the source code in question, plus the positive or negative impact in the overall complexity, results in the score reached by the developer.

After collecting the score, it is transferred to the gamification server, which will include a record of activity and scores in the developer's profile. Developers then gain access to a web system that will display the leaderboard, displaying their performance.

Furthermore, a number of rules can also be included in the server with specific scores to improve the developer's motivation; for example, an additional 100 points every 25 commits or additional 10 points every time are given he or she reduces the overall complexity.

It is up to the administrator to set up the leaderboard to identify the developers or to keep them anonymous, since the effects of anxiety due to are competition yet to be studied. In software development, it is common for developers to work individually and in isolation, which can lead to dissatisfaction, boredom and eventually decreased productivity and even quality. A source management server can create a perception of collaboration in software development.

8 Conclusion

Measuring developers' collaboration or their engagement in a software development project by measuring only the number of produced lines has already proven to have limited effectiveness, as concluded in previous researches.

Calculating the engagement of a developer using a metric that tends to be more assertive than those presented in previous studies, is the goal of this paper. Therefore, we presented the equation that combines the McCabe's cyclomatic complexity with the number of lines of code changed by the developer to obtain a balanced score between simple and complex codes.

This also prevents situations caused by a developer who wishes to rise in his or her position, by starting to make changes in several lines of simple code to receive a higher score than someone who is working on a complex code.

With McCabe's metric, we counterbalanced the effect generated by Pareto's principle, preventing developers that work in 80% of code, which possesses 20% of all the complexity, to gain an additional advantage in the ranking proposed by gamification. The development process is thus more collaborative because it balaces the accomplished work and the scores achieved tend to be more balanced.

To give sequence to this research, the implementation of the proposed equation and the prototype tool are still necessary. The proposed model must, therefore, be applied to an actual development project to validate the proposed theory and measure any gains of productivity and engagement in the project.

References

1. Ahmad, I., Jaafar, A.: Games design and integration with user's emotion. In: 2011 International Conference on User Science and Engineering (i-USEr), pp. 69–72 (2011)

2. Chidamber, S.R., Darcy, D.P., Kemerer, C.F.: Managerial use of metrics for object-oriented software: An exploratory analysis. IEEE Transactions on Software Engineering 24(8), 629–639 (1998)
3. Cook, D.: What are game mechanics. Lost Garden (2006), `http://www.lostgarden.com/2006_10_01_archive.html`
4. Crawford, C.: Chris Crawford on game design. New Riders (2003)
5. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining gamification. In: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, pp. 9–15. ACM (2011)
6. Dubois, D.J., Tamburrelli, G.: Understanding gamification mechanisms for software development. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, pp. 659–662. ACM (2013)
7. Groh, F.: Gamification: State of the art definition and utilization, pp. 39–47. Institute of Media Informatics Ulm University (2012)
8. Hugos, M.: Enterprise Games: Using Game Mechanics to Build a Better Business. O'Reilly (2012)
9. Koc, A., Tansel, A.: A survey of version control systems. In: ICEME 2011 (2011)
10. Koster, R.: Theory of fun for game design. O'Reilly Media, Inc. (2010)
11. Louridas, P., Spinellis, D., Vlachos, V.: Power laws in software. ACM Transactions on Software Engineering and Methodology (TOSEM) 18(1), 2 (2008)
12. Marczewski, A.: Gamification: A Simple Introduction. Andrzej Marczewski (2012)
13. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering (4), 308–320 (1976)
14. O'Grady, S.: The New Kingmakers. O'Reilly Media (2013)
15. Romero, M., Usart, M., Ott, M., Earp, J.: Learning through playing for or against each other? Promoting collaborative learning in digital game based learning. Learning 5, 15–2012 (2012)
16. Rosenberg, J.: Some misconceptions about lines of code. In: Proceedings of the Fourth International Software Metrics Symposium, pp. 137–142. IEEE (1997)
17. Salen, K.: Rules of play: Game design fundamentals. The MIT Press (2004)
18. Singer, L., Schneider, K.: It was a bit of a race: Gamification of version control. In: 2012 2nd International Workshop on Games and Software Engineering (GAS), pp. 5–8. IEEE (2012)
19. Toth, Z., Novak, G., Ferenc, R., Siket, I.: Using version control history to follow the changes of source code elements. In: 2013 17th European Conference on Software Maintenance and Reengineering (CSMR), pp. 319–322. IEEE (2013)
20. Treude, C., Storey, M.: Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering, vol. 1, pp. 365–374. IEEE (2010)
21. Velazquez, A., Martinez-Garcia, A.I., Favela, J., Hernandez, A., Ochoa, S.F.: Design of exergames with the collaborative participation of older adults. In: 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 521–526. IEEE (2013)
22. Watson, A.H., McCabe, T.J., Wallace, D.R.: Structured testing: A testing methodology using the cyclomatic complexity metric. NIST special Publication 500(235), 1–114 (1996)
23. Zichermann, G., Cunningham, C.: Gamification by Design: Implementing game mechanics in web and mobile apps. O'Reilly Media, Inc. (2011)