It Was a Bit of a Race: Gamification of Version Control

Leif Singer, Kurt Schneider
Software Engineering Group
Leibniz Universität Hannover
Hannover, Germany
{leif.singer, kurt.schneider}@inf.uni-hannover.de

***Abstract*—The adoption of software engineering practices cannot always be achieved by education or processes. However, social software has the potential for supporting deliberate behavior change. We present preliminary results of an experiment in which we encouraged computer science students to make more frequent commits to version control by using a social software application. We provided a web-based newsfeed of commits that also displayed a leaderboard. While we have yet to analyze the data, interviews we conducted with the participants allow for first qualitative insights.**

Version control; commits; behavior change; gamification

I. Introduction

Because of deficits in motivation [1] or expertise, developers don't always strictly follow processes and software engineering best practices. We believe using social software can mitigate this problem by addressing the intrinsic motivations of developers. In this paper, we use version control as an example of a software engineering method for which we want to improve the adoption of best practices. More specifically, we are concerned with the frequency of commits to repositories.

To make changes to a software product easily reversible or to select them for releases, it is helpful to make commits that are thematically cohesive and isolated from other changes. Committing frequently can be a prerequisite for this. Figure 1 shows a commit to a repository hosted on the GitHub social coding website. As he did not commit frequently enough, the author is unable to tell which changes the commit contains.

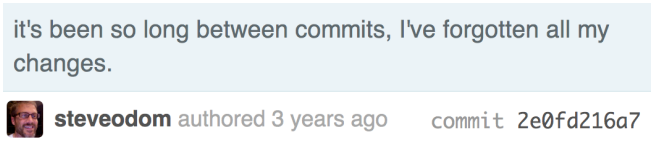

Figure 1. A commit on GitHub[1]

While there are situations in which other commit strategies might be preferable, our current work concentrates on this strategy and having it adopted by developers. We often find this in student projects: students put several different features and fixes into a single commit. We believe the reason to be a combination of missing knowledge regarding best practices and the effort needed for thoughtful commits.

Social software allows people to connect and interact with each other and to stay aware of what their contacts are doing. This has been successfully used in software engineering already, as Treude and Storey [12] as well as Begel and Zimmermann [2] have documented, for example. They show that newsfeeds can be used to increase the awareness of project participants.

However, social software can also be used to motivate users and to influence their behavior. Centola showed that social reinforcement can make behavior spread in an online social network [3]. Foster et al. used a custom Facebook application and step counters to show that social software can even motivate people to walk more [6].

These mechanisms are related to gamification, which, according to Deterding et al. [5], is "the use of game design elements in non-game contexts." Gamification can be used to motivate people regarding certain tasks. For example, Thom et al. discuss a company-internal social network site, in which users were awarded points and ranks for contributing [11]. Removing these game mechanics resulted in a drop of contributions. Landers and Callan used similar mechanisms for encouraging students to take non-mandatory quizzes [7].

We want to use these mechanisms in a systematic manner to improve the adoption of software engineering practices among developers [10]. This strategy would complement – not replace – existing approaches from developer education and software processes. In this paper, we describe how we used our approach to encourage computer science students to make commits to version control more frequently. We use the number of commits of a developer as a very crude metric for commit quality.

II. Experiment Setup

Each fall semester, our research group organizes the *software project* course, a mandatory course for computer science undergraduates. The course has about 35 to 60 participants each semester, most of them in their fifth semester. The students form teams of four to six and elect a project leader and a quality agent. The project starts at the beginning of October and lasts until the end of January. In the term we ran our experiment, we had 37 students.

The members of our group act as customers: we propose software that we would like to have developed and have the students elicit requirements from us. In the second phase, the teams can choose between preparing an architecture or

[1] Source: https://github.com/steveodom/beta-signup/commits/master/views, accessed Feb 9th, 2012

creating exploratory prototypes. They implement the actual applications in the third and final phase. During the project, a member of our group will act as coach, answering questions about technical subjects and the development process. To create time pressure, each team receives six vouchers for customer appointments of 15 minutes each and six vouchers for coach appointments of 30 minutes each.

At the end of the project, the customer executes the acceptance tests defined in the previously created requirements specification and decides whether reworking is needed. At this point, our role-play ends.

Finally, we conduct an LID session with each team. LID – short for *Light-weight Documentation of Experiences* – is a technique for the elicitation of project experiences [9]. A typical LID session for the course takes about two hours during which the team members and a moderator jointly fill in a template for experience elicitation. It inquires students about impressions, feelings, conflicts, and advice, and makes them review the whole project from beginning to end. In the sessions, we emphasize that the passing of the course will not be affected anymore and encourage them to honestly describe the negative experiences as well.

For each team, we provide a Subversion repository, a Trac instance for issue tracking, and a web-based quality gate system that is used to progress the teams through the project phases. The Trac instance is linked to the team's repository, so students are able to see their team's commits using either Trac or any subversion client. In the fall 2011 course, we added another tool: a web-based newsfeed of each team's commits, featuring a leaderboard that shows the commit count for each team member. The next section introduces *Teamfeed*, a web-based newsfeed of commits.

III. A Newsfeed of Commits to Version Control

The *Teamfeed* web application periodically reads the commits to each team's repository and saves them to a database. They are then displayed in a newsfeed for each team. Every student in the project could log in to Teamfeed using their Subversion account and was then presented with their respective team's newsfeed. The newsfeeds of other teams were not accessible to the students. Figure 2 shows an anonymized screenshot of the application in which the names of students and their team have been altered. Several other texts have been translated into English.

Students could comment on posts in the newsfeed. Comments on a student's own commit or on a discussion in which the student was already participating resulted in an email notification being sent to the student. If a student had uploaded an image for use as their avatar, it was displayed next to their commits; otherwise a default image was used.

Additional posts to the newsfeed were generated when predefined thresholds regarding the number of commits by a user or a team were exceeded. We called these posts *milestones* in Teamfeed. We defined thresholds of 1, 10, 25, 50, 100, 250, 500, 750, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7500, and 10000 commits. These generated posts such as "Congratulations! Jane Doe has reached her 200th commit!" or "Wonderful! Your team has just reached the 1000th commit!" We based the thresholds on previous semesters' commit counts and added a buffer. Similar to comments, reaching a milestone also triggered an email notification to be sent to either the student or the whole team.

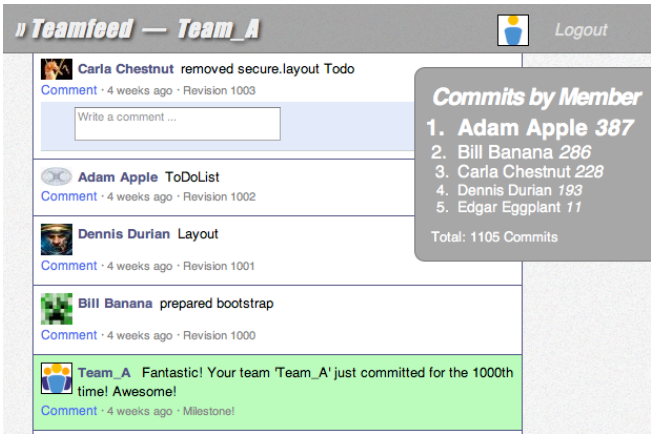

Figure 2. A screenshot of Teamfeed's newsfeed and leaderboard.

On the right, a leaderboard lists the team members and the counts of their respective commits so far. For higher ranks, name and commit count were displayed larger.

Each Sunday at around 3pm, Teamfeed sent out a weekly digest to each student such as the one depicted in Fig. 3. The digest summarized how many commits the individual student had made in the past week, but also provided this information about their teammates. It also mentioned milestones that were reached during the week and showed the current state of the leaderboard.

```
Hello Edgar Eggplant!

In this weekly digest, we report what you and your team
achieved in the past seven days.

You made 18 commits this weeks.

Your teammates achieved the following:
* Adam Apple made 36 commits this week.
* Dennis Durian made 10 commits this week.
* Carla Chestnut made 32 commits this week.
* Bill Banana made 21 commits this week and, in doing so,
reached a milestone: the 250th commit.

In total, your team made 117 commits this week. This
allowed you to reach a milestone: your 1000th commit.
Excellent!

This is the current status of your team:
        1. Adam Apple (387)
        2. Bill Banana (286)
        3. Carla Chestnut (228)
        4. Dennis Durian (193)
        5. Edgar Eggplant (11)

Best wishes for next week!
```

Figure 3. A weekly digest as sent by Teamfeed.

A. Rationales

We now present the rationales for implementing the aforementioned elements in Teamfeed.

Newsfeed: Newsfeeds can be used to improve the awareness of project members. See, for example, Treude's

and Storey's [12] or Begel's and Zimmermann's work [2]. Also, they can help spreading the displayed behavior [3].

Commenting: Foster et al. conducted an experiment that indicates a positive influence of discussion on the adoption of behavior [6].

Notifications: Notifications about interesting and positive events are an integral part of social software. For example, Facebook and Twitter send out emails to users when others want to connect or already connected with the user, or when others react to the user's content. In our view, these notifications encourage users to use the application more and to get into contact with others.

Milestones: Psychology research in goal setting and task motivation found that defining specific goals has several benefits [8]. Defined goals direct attention to goal-relevant tasks, in our case committing to version control. By generating milestones, we implicitly set those goals. After a few commits, participants realized that the next milestone would be further off – we deliberately increased the spacing between successive milestones. As high goals were found to lead to increased effort compared to lower goals, this spacing should have been effective in increasing the effort spent on creating commits. Complex goals may have the adverse effect, though: they can lead to performance anxiety and pressure. In line with these results, we created some early milestones that were easy to reach and accustomed the students with committing to version control. The effects that can be achieved may differ wildly between different settings, however, cannot be guaranteed, and need to be implemented with care.

Leaderboard: Thom et al. showed that awarding points and ranks for contributions was effective in motivating users to contribute content to a company-internal social network site [11]. Again, these approaches must be viewed critically – some individuals might prefer earning points over the task itself; others could reject the competitive situation and become demotivated.

Weekly digest: Social websites such as Quora and LinkedIn periodically send out digest emails. These document what content was contributed by the user's contacts. This inspired us to create the weekly digest, as we believe that this is another mechanism that reminds and encourages users to use the application again.

IV. Preliminary Results

In the LID sessions of the 2011 fall course, we asked the students about the web-based newsfeed of version control commits. This section presents the results from these questions. They give interesting insights into the way the students perceived *Teamfeed* and their own commit behavior. As the LID technique by design is not overly precise, we cannot present any exact numbers regarding the students' statements. Rough trends are visible, however.

A. Negative statements

Some of the participants were unable to tell what Teamfeed was, even though they had regularly received the weekly digest emails and notifications of reached milestones. As the Trac issue tracker was also connected to the Subversion repository, these students used Trac instead of Teamfeed.

Many students perceived the emails as spam. The emails congratulating them on reaching milestones were mostly sneered at.

The metric we used – the number of commits by a person – was often said to be too simplistic and useless. Just because someone had submitted something to version control, they said, did not mean that there was actually any value in the commit. One of the students was unable to access the internet on weekends and thus was unable to commit to the centralized subversion repository. On Mondays, the student would commit all changes from the weekend in a single commit.

Finally, some of the students criticized that the files affected by a commit were not visible from the application. Also, long commit messages were cut off at 140 characters, which was mentioned as a nuisance by several students.

B. Positive statements

Many students mentioned that the application was useful for getting an overview of their project. A few even mentioned that they explicitly looked at the application to find out whether a commit they were waiting for from another student had already occurred. Repeatedly, the relative simplicity compared to Trac was mentioned as an advantage, for example when accessing it on the go using a smartphone.

Similar things were said about the weekly digest: *"It gave a quick overview. And it was sent often enough, but not too often."* Additionally, the members of one team explicitly mentioned motivation: *"It motivates you because you see things moving forward. You see progress."*

Several students also saw the limitations of our simplistic metric, but at the same time mentioned that it helped them anyway. They mentioned positively that the leaderboard and, partly also the newsfeed, allowed them to see whether someone was not participating all that much. Related to that, they said they did not want to be last on the leaderboard and admitted a certain motivational effect.

During the LID sessions, some students told us about sarcastic remarks they made to each other during the project, such as *"come on, you're just fixing that bug to gain a commit!"* We believe this shows that even though the leaderboard was not taken very seriously, it indeed *was* on the minds of the students.

When a student criticized the competitive nature of the leaderboard and, pointing at two teammates, mentioned that *"it was a bit of a race between the both of you"*, one of the students he was talking about chimed in, saying, *"... which I won!"* This might indicate that while the competitive situation was not comfortable, it may have been effective anyway.

One student was especially critical of the whole endeavor and said the application had encouraged him to make superfluous commits – which he considered a bad thing. When we followed up on that, he told us, *"I committed things earlier rather than later. It felt like I tried to game the system. I mean, the commits did contain actual*

changes, but … when I was at 90 commits I preferred to make several smaller commits to reach the milestone I expected at 100. Actually, I'm not sure if that's a bad or a good thing, to keep the commits so small. It made me commit small fixes immediately instead of committing everything at once after an hour of programming."

The behavior we see in the last quote was exactly what we were hoping for. Even though the student had thought that large commits were preferable, our application motivated him to make *more* and *smaller* commits.

V. Conclusions & Outlook

We presented an approach to using mechanisms of social software to gamify version control. After 37 students used our application in a semester-long cooperative development project, we used the LID technique to interview them about their experiences. We found a balance of positive and negative comments. While the weekly emails might have annoyed some of the students, they nevertheless kept many students aware of what their teammates were doing. In a few instances, we found that we were able to evoke the exact responses we had been hoping for: students making more and smaller commits. Even though such approaches can backfire, we believe it is important to research these mechanisms further.

Depending on the software development process employed, the culture, and the goals of the developerment organization, there are other valid strategies for version control. For example, one organization might want to improve the relationship of commits to work items from an issue tracking system and design a game system around that goal. In other organizations, it might be more important to assess the actual quality of commit messages, which might be achieved by having developers anonymously rate their peers messages and base a point system on these ratings. The *Continuous Integration Game plugin* for the Jenkins continuous integration tool awards points based on commits that keep the repository in a compilable state or for adding unit tests. Using additional rules, it can even be configured to award points based on reports from tools such as FindBugs and Checkstyle[2].

While obviously several point-based systems are imaginable – some even with a social rating component – we believe they should only be used for comparably routine tasks, such as committing more often. On creative tasks, however, extrinsic rewards such as points can even have a detrimental effect [4]. They should be supported by a system that clears the way for the more intrinsic motivations of developers, which we will discuss in future work.

Our next step will be to analyze the commit data and compare it with the previous years of software projects. Since the process and the kinds of projects are very similar each year, we believe we might be able to extract proof as to whether our approach was really effective.

This work is part of our efforts to create a method for systematically applying mechanisms of behavior change to software engineering [10]. Our aim is to improve the adoption of best practices by software developers. While processes and education play an important role in this regard, we believe our approach can be a valuable addition to the options software development companies have at their disposal.

Acknowledgment

We thank Christoph Treude, Sebastian Meyer, and Kai Stapel for their helpful comments on a draft of this paper. We're grateful to our reviewers for their insightful additions.

References

[1] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. Motivation in Software Engineering: A systematic literature review. *Information and Software Technology*, 50(9-10):860–878, 2008.

[2] Andrew Begel and Thomas Zimmermann. Keeping Up With Your Friends: Function Foo, Library Bar.DLL, and Work Item 24. In *Web2SE: First Workshop on Web 2.0 for Software Engineering (co-located with ICSE 2010)*, 2010.

[3] D. Centola. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194, 2010.

[4] E.L. Deci and R.M. Ryan. *Handbook of self-determination research*. The University of Rochester Press, 2002.

[5] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of MindTrek'11*. ACM, 2011.

[6] D. Foster, C. Linehan, B. Kirman, S. Lawson, and G. James. Motivating physical activity at work: using persuasive social media for competitive step counting. In *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 111–116. ACM, 2010.

[7] R.N. Landers and R.C. Callan. Casual Social Games as Serious Games: The Psychology of Gamification in Undergraduate Education and Employee Training. *Serious Games and Edutainment Applications*, pages 399–423, 2011.

[8] Edwin A. Locke and Gary P. Latham. Building a practically useful theory of goal setting and task motivation: A 35-year odyssey. *American Psychologist*, 57(9):705, 2002.

[9] Kurt Schneider. LIDs: A Light-Weight Approach to Experience Elicitation and Reuse. In Frank Bomarius and Markku Oivo, editors, *Product Focused Software Process Improvement*, volume 1840/2000 of *Lecture Notes in Computer Science*, pages 407–424. Springer Berlin / Heidelberg, 2000.

[10] Leif Singer and Kurt Schneider. Influencing the Adoption of Software Engineering Methods using Social Software. In *34th International Conference on Software Engineering (ICSE), NIER Track (in press)*, 2012.

[11] Jennifer Thom, David R. Millen, and Joan DiMicco. Removing Gamification from an Enterprise SNS. In *Proceedings of the 2012 ACM Conference on Computer Supported Cooperative Work*. ACM, 2012.

[12] C. Treude and M.A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, 365–374. ACM, 2010.

[2] Source: https://wiki.jenkins-ci.org/display/JENKINS/The+Continuous+Integration+Game+plugin, accessed Mar 27th, 2012