LETTER *Special Section on Empirical Software Engineering*

How Does Time Conscious Rule of Gamification Affect Coding and Review?*

Kohei YOSHIGAMI[†], **Taishi HAYASHI**[††], ***Nonmembers***, **Masateru TSUNODA**[††a)], **Hidetake UWANO**[†††], ***Members***, **Shunichiro SASAKI**[††], ***Nonmember***, ***and*** **Kenichi MATSUMOTO**[†], ***Member***

SUMMARY Recently, many studies have applied gamification to software engineering education and software development to enhance work results. Gamification is defined as "the use of game design elements in non-game contexts." When applying gamification, we make various game rules, such as a time limit. However, it is not clear whether the rule affects working time or not. For example, if we apply a time limit to impatient developers, the working time may become shorter, but the rule may negatively affect because of pressure for time. In this study, we analyze with subjective experiments whether the rules affects work results such as working time. Our experimental results suggest that for the coding tasks, working time was shortened when we applied a rule that made developers aware of working time by showing elapsed time.

key words: *human factors, motivation, work efficiency, gamification*

1. Introduction

In software development, one of the most important objectives is work efficiency. To achieve this objective, gamification has recently been applied to activities related to software development. Gamification is defined as "the use of game design elements in non-game contexts" [2]. The aim of gamification is to enhance motivation, and the outcome of work.

However, the effects of gamification on software development have not been sufficiently evaluated. For example, in studies of gamification related to software development, approximately half did not evaluate the effects of gamification experimentally [6]. In addition, as far as we know, few studies evaluated gamification based on the working time of subjects, although quantitative analysis was performed in some studies.

When applying gamification to software development, we create various rules. However, the effect of rules is uncertain. For example, if we create a time limit for work as a rule of gamification, it may shorten the working time. In contrast, it may not shorten the time because of pressure for time. Additionally, if we apply a time limit to a code review, it may shorten review time. However, it is possible that the number of faults found decreases. Therefore, we focus mainly on whether the rules of gamification shorten working time or not. It would be helpful to consider how to apply the rules of gamification by clarifying the influence of those rules.

The major contribution of our study is that using subjects, we evaluated effect of gamification based on working time. About half of the studies did not evaluate gamification experimentally, and only few studies have evaluated the effect of gamification on working time.

2. Gamification

Gamification aims to enhance the motivation of workers by adding amusement factors to the work process. In recent years, this approach has attracted attention in various fields [3], [7]. For example, gamification is represented in the following approaches [6]:

- **Point**: A worker gets points when a task is finished.
- **Ranking**: Based on the points, higher-ranked workers are shown to other workers.
- **Badge**: A worker gets various badges (in the system) on finishing a task.
- **Level**: Workers' level goes up when the point level reaches a certain criterion.

The effect of getting items and badges is that it stimulates the desire to collect, and this is expected to enhance motivation for tasks. Also, by showing the points accumulated, workers can compete with themselves and with others, and this may spur motivation.

Pedreira et al. [6] mapped studies of gamification in the software engineering field based on a target development process, gamification approaches, and evaluation of the gamification effect. Approximately half of the studies did not evaluate gamification experimentally, and many studies demonstrated new proposals only.

To evaluate the effect of gamification, many studies have focused on cumulative work results, such as the number of faults found and the number of commits. Few studies have evaluated the effect of gamification on working time. Khandelwal et al. [4] applied gamification to code review,

Manuscript received February 28, 2019.
Manuscript revised July 1, 2019.
Manuscript publicized September 18, 2019.

[†]The authors are with Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The authors are with Kindai University, Higashiosaka-shi, 577–8502 Japan.
[†††]The author is with National Institute of Technology, Nara College, Yamatokoriyama-shi, 639–1080 Japan.

*This work is an extended study of M. Tsunoda et al., "How Do Gamification Rules and Personal Preferences Affect Coding?" Proc. of International Workshop on Empirical Software Engineering in Practice (IWESEP), pp.13–18, 2018.

a) E-mail: tsunoda@info.kindai.ac.jp

DOI: 10.1587/transinf.2019MPL0002

and analyzed the difference in working time when gamification was applied. However, as far as we know, no study has quantitatively analyzed whether working time is shortened when a rule is changed.

3. Experiment

In the experiment, subjects performed a code review task and a coding task sequentially. In the review task, subjects found faults in source code, and on the coding task, they created programs according to specifications. After the coding task, subjects answered a questionnaire. The coding task was performed one week or more after the review task. Before the experiment, we told subjects that they would know their rank after the experiment to stimulate a sense of competition within the subjects. Subjects were undergraduate students who majored in information science. The number of subjects on the review task was 13, and the number on the coding task was 14.

We set the rules of gamification considering the following aims:

- The aim to shorten the time for finishing tasks related to software development. For this aim, we applied rules that made subjects aware of working time. We call it *time conscious rule*.
- The aim to enhance the effects of gamification (i.e., enhance the motivation of participants). For this aim, we applied a rule to analyze the effects of gamification that had continuity (i.e., the previous results, such as score and ranking, affect subsequent results). We call it *continuity rule*.

Note that we did not evaluate all aspects of gamification rules, but we mainly focused on rules which are expected to affect to working time.

3.1 Code Review Task

On the code review task, we showed subjects specifications of the programs (including no faults) and source code (including faults), and the subjects indicated found faults and how to modify them. The source code was written in Java, which all subjects understood. To analyze the negative effect of gamification rules, we focused on the number of found faults.

We selected two review targets, illustrated in study [5]:

- Target A: A program outputs a sum of numbers. The length of the specification document is about 50 words, and the number of lines of source code is about 200. The source code includes five faults.
- Target B: A program reads a file and finds target data. The length of the specification document is about 300 words, and number of lines of source code is about 130. The source code includes eight faults.

First, subjects reviewed target A, and after that, they reviewed target B. We made the following gamification rules

Class	Line	Faults
sum	7	Not i--, but i++
main	13	Not m, but n
sum	3	Not sum=1, but sum=0

Score	Avg.	Score per fault
1500	1000	200

Score = 1200 (given score) - 300 (elapsed seconds) + 200 * 3 (# of faults)

Fig. 1 Screen images on review task, applying rule β

α and β:

- Rule α: The score is calculated based on the number of found faults.
- Rule β: The score is calculated based on the number of found faults and the review time. The rule is time conscious rule.

For each found fault, we added 200 points for target A, and 250 points for target B, based on a preliminary experiment. When we applied rule β, a certain score was given first, and 5 points were deducted for every 5 s elapsed during the review. The given score for target A was 1200, and it was 3600 for target B. The score increased when faults were found, although elapsed time reduced the score.

The score was shown during the task. We increased the score when found faults were input, even if the indications were incorrect. When subjects were ranked, the score was modified if the indications were incorrect. The modification was explained to subjects before the experiment.

When applying rule β, we show the average score (based on a preliminary experiment) to enhance the motivation of the subjects. In actual software development, such a target score can be shown based on past data. Figure 1 shows the screen images that were shown to subjects in the experiment. Figure 1 shows that the subjects wrote details of the found faults in the upper table. The lower table shows the score, average score (shown only when rule β was applied), and the added points for each found fault. We note that the figure does not show the specifications and source code that the subjects read during the experiments.

We created two groups and changed the combinations of review targets and gamification rules.

- Group 1: Rule β was applied to review target A, and rule α was applied to review target B (The number of subjects was six).
- Group 2: Rule α was applied to review target A, and rule β was applied to review target B (The number of subjects was seven).

3.2 Coding Task

On the coding task, subjects created programs based on

given specifications. Java was used as the programming language because all subjects understood it. Programs were made with Eclipse, and when the execution results were correct, the tasks were considered finished (i.e., the tasks were regarded as completed when the output texts by the program were the same as the correct texts, which were shown in the specifications). We regarded the quality of the tasks as sufficient when the results were correct, and we did not evaluate the quality of the created source code using quantitative metrics. In actual software development, software testing is often strictly performed. In contract, source code is generally not modified based on the metric evaluation after coding is finished.

We gave subjects the following specifications. We selected those that were relatively easy from a web site (AIZU ONLINE JUDGE http://judge.u-aizu.ac.jp/):

- Specification C: output the multiplication table
- Specification D: read 10 values, and output three of them in descending order

First, subjects created a program based on specification C, and after that, they created one based on specification D.

We made the following gamification rules:

- Rule γ: Score (i.e., elapsed time) is not shown on the tasks. The score, and the ranking based on the score are shown after the tasks are finished.
- Rule δ: Score (i.e., elapsed time) is shown on the tasks. The sum of scores on the coding and review task (see Sect. 3.1) and the ranking are shown after the tasks are finished. The rule includes the time conscious rule and the continuity rule.

Figure 2 shows a screen capture when the rule is applied. The left text box in Fig. 2 shows the specifications, and the upper right text explains the continuity rule (shown only when rule δ was applied). The middle right box shows the score (shown only when rule δ was applied). The lower right button was pushed when the task was completed.

For each rule, subjects were given a certain score, and 5 points were deducted for every 5 seconds elapsed during the task. We created two groups, and changed the combinations of coding specifications and gamification rules.

- Group 3: Apply rule δ on specification C, and apply rule γ on specification D (The number of subjects was nine).

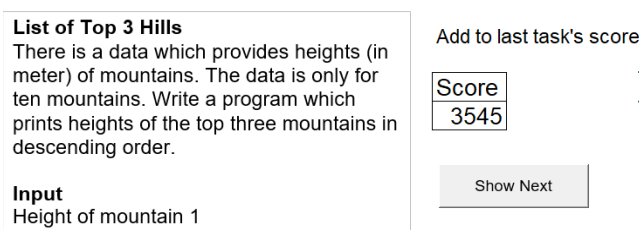

Fig. 2 Screen image on coding task, when applying rule δ (showing elapsed time)

- Group 4: Apply rule γ on specification C, and apply rule δ on specification D (The number of subjects was five).

3.3 Subjective Evaluation

We collected subjects' evaluation of gamification with the following questionnaire after the coding task. Subjects answered questions about their feelings on a five-point scale (1: strongly disagree ... 5: strongly agree).

- Q1: Did you feel that gamification affected your work efficiency and motivation?
- Q2: Did you feel that showing the score affected your motivation?
- Q3: Did you feel that adding the previous task's score (i.e., review score) affected your motivation?

We did not ask whether subjects liked code review or coding, because we only relatively compared results of applying rules on each task (i.e., review and coding tasks).

4. Relationships between Rules and Work Outcomes

To clarify the point of the analysis, we set the following research questions:

- **RQ1**: For code review, when we applied a rule that made subjects aware of working time, does it affect the review time and the number of found faults?
- **RQ2**: For the coding tasks, is working time shortened when we applied a rule that made developers aware of working time?
- **RQ3**: Is the evaluation of gamification rules was different between individuals?

4.1 Influence on Code Review

To answer RQ1, we analyzed review time and the number of found faults on code review.

Review time: Review time is shown in Table 1. Bold face indicates cases where rule β (considering elapsed time) was applied. If rule β shortens the review time, then the review time for A is expected to be comparatively shorter for group 1, and the review time for B is also expected to be shorter for group 2. To analyze this effect, we defined a review time ratio, which is review time of B divided by review time of A. The ratio is expected to be larger for group 1 than for group 2 if rule β shortens review time. However, as shown in the table, the difference in the ratio between group 1 and group 2 was very small. Therefore, rule β is not considered to shorten review time.

Table 1 Difference in review time by rules (Bold: considering elapsed time)

Group	Target A (a)	Target B (b)	Time ratio (b / a) Average	Time ratio (b / a) Standard deviation
1	**0:11**	0:22	2.1	0.4
2	0:15	**0:29**	1.9	0.7

Table 2 Difference in found faults by rules (Bold: considering elapsed time)

Group	Target A (a)	Target B (b)	Found fault ratio (b / a) Average	Found fault ratio (b / a) Standard deviation
1	**3.2**	4.0	1.3	0.5
2	2.7	**3.7**	1.5	1.1

Table 3 Difference in coding time by rules (Bold: showing elapsed time and adding review score)

Group	Specification C (c)	Specification D (d)	Time ratio (d / c) Average	Time ratio (d / c) Standard deviation
3	**0:09**	0:34	4.0	1.3
4	0:12	**0:31**	2.9	1.2

Found faults: Table 2 shows the number of found faults on the review. Bold face indicates cases where rule β was applied. We defined found fault rate as with the review time ratio. The ratio is expected to be larger for group 1 and smaller for group 2 if rule β decreases the found faults. As shown in the table, the ratio of group 1 was almost same as the group 2. Hence, rule β did not affect the number of found faults.

The result suggests that when we apply a gamification rule that makes workers aware of working time during code review, the rule should not be expected to shorten review time very much. This is because workers must read through source code to review it, and their reading speed may not change dramatically. Also, the rule did not negatively affect found faults. So, there is no need to apply a rule that emphasizes working time during code review, and the answer of RQ1 is "No."

4.2 Influence on Coding

To answer RQ2, we analyzed coding time. Table 3 shows the coding time elapsed during the experiment. Bold face indicates cases where rule δ (showing elapsed time and adding the review score) was applied. We defined a coding time ratio where coding time based on specification C was the denominator and the time based on specification D was the numerator. If rule δ shortens coding time, the time based on specification C (i.e., the denominator of the ratio) is expected to be comparatively shorter for group 3, and the time is expected to be longer for group 4. In this case, the ratio of group 3 is larger than that of group 4.

As shown in the table, the ratio of group 1 was larger. When we applied a statistical test, difference of the ratio between the groups was not significant (the p-value was 0.12). However, considering the number of data points, there may be type II error. Actuarially, when we duplicated data by copying (i.e., the number of data point was 28), the p-value became 0.02. To check the difference visually, we made the boxplot as shown in Fig. 3. As shown in the figure, median and the position of the boxes (i.e., first quartile and third quartile) is very different.

Therefore, rule δ is considered to shortened the coding time. Although rule δ showed elapsed time and added the review score, showing the time is considered to be more effective in shortening the working time. This is because showing time was evaluated highly on Q2 (the details of the analysis are explained in Sect. 4.3). So, the answer of RQ2 is "Yes." Note that the influence of the rule on coding approach (i.e., coding quality) is not clear, even if the execution results of the program are correct. Thus, further analysis of the influence is needed.

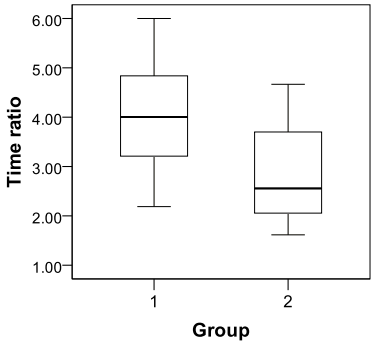

Fig. 3 Boxplot of time ratio stratified by the group

Table 4 Years of programing experience of subjects

Group	Task	Average	Median	Standard deviation
1	Review	4.1	3	3.0
2		3.3	3	0.5
3	Coding	3.3	3	0.5
4		4.1	3	2.7

Table 5 Correlation coefficients to years of programming experience of each subject

	Review time ratio	Found fault ratio	Coding time ratio
Correlation coefficient	-0.10	-0.23	-0.07
p-value	0.74	0.45	0.83

4.3 Influence of the Subject Characteristics

The results presented in Sects. 4.1 and 4.2 could depend on the characteristics of the subjects, such as their programming level or programming experience. However, precisely measuring their programming level is not easy. Thus, we only considered their years of programing experience as one of the characteristics and analyzed the influence on the results. The summary statistics on the years of programing experience in each group are listed in Table 4. In Groups 1 and 4, the number of years of experience of a subject was 10, which affected the average and standard deviation of the groups. Except for the subject the number of years of experience was between two and four, i.e., the programming experience among the subjects did not greatly differ.

Table 5 lists the relationship between the number of years of experience and the ratios of the review time, discovered fault, and coding time. We used the Spearman's rank correlation coefficient to avoid the influence of outliers. Table 5 indicates that the relationships were weak (the correlation coefficients were smaller than 0.3).

Additionally, we divided the subjects into two groups

Table 6 Difference in review time by years of experience

Group	Target A (a)	Target B (b)	Time ratio (b / a) Average	Time ratio (b / a) Standard deviation
Short	0:13	0:24	1.9	0.6
Long	0:14	0:28	2.1	0.6

Table 7 Difference in found faults by years of experience

Group	Target A (a)	Target B (b)	Found fault ratio (b / a) Average	Found fault ratio (b / a) Standard deviation
Short	3.0	4.0	1.5	1.1
Long	2.8	3.7	1.3	0.6

Table 8 Difference in coding time by years of experience

Group	Specification C (c)	Specification D (d)	Time ratio (d / c) Average	Time ratio (d / c) Standard deviation
Short	0:12	0:38	3.5	1.4
Long	0:08	0:26	3.5	1.4

Table 9 Basic statistics of the questionnaires

	Q1	Q2	Q3
Average	3.5	3.7	3.1
Standard deviation	1.3	1.0	1.3
Median	4	4	3
Ratio of low evaluation (%)	29	14	43

(i.e., whether the number of years of experience is short or long) using the median of the number of years of experience. The number of subjects in each group was almost the same. We created Tables 6–8, and their structures were almost the same as those in Tables 1–3 except for the grouping factor. Tables 6 and 7 illustrate that the review-time and discovered-fault ratios were almost the same, and the results were similar to those listed in Tables 1 and 2. In Table 8, although the coding time of the "long group" was shorter than that of the "short group," the coding ratio between them was almost the same. This result was different from that listed in Table 3, which suggested that the result listed in Table 3 was not affected by the number of years of experience. Therefore, the results in Sects. 4.1 and 4.2 are not considered to be affected by the programming experience, which is a major characteristics of the subjects.

4.4 Preference for Gamification

To answer RQ3, we analyzed preference for gamification. In addition, we analyzed whether the last result of gamification affects the preference or not.

Preference for gamification: We analyzed to what extent subjects liked gamification and its rules. Table 9 shows the average and standard deviation of the answers to Q1-Q3, and the ratio of answers that were lower than three (i.e., the ratio of low evaluations). On Q1 and Q2, the average was larger than 3.5, and the ratio was not large. Therefore, the evaluations were relatively high. In contrast, the ratio was 43% for Q3, and therefore the evaluation of the rule was different among subjects.

Influence of the last result on preference: When the last result of gamification, i.e., the result of gamification on the review task, is good, subjects may like gamification more. To clarify this influence, we analyzed the relationship of the last result with Q1 and Q3. The results showed that each correlation coefficient was very small (The correlations were 0.07 and 0.01). That is, the preference for gamification and its rules were not affected by the last result. So, based on the above results, the answer of RQ3 is "Yes."

4.5 Discussion

The effect of the gamification rules could change when the experiment time and duration increases. For example, when the experiment time is long, the subjects turn to gamification, and the effect might be larger. This topic would be a future work that could be done to analyze the relationship between the experiment time and the effect of the gamification rules. We note that our experiment time is not very short to obtain this effect compared with that in previous studies. For example, Arai et al. [1] applied gamification to remove the warnings made by static analysis tools. In their experiment, the subjects coped with two tasks, and each task was performed within 30 min. As a result, they successfully observed the gamification effects.

We used rankings as a reward for gamification. The effect of the gamification rules could change if other rewards are used, such as badges. This would also be a future topic to analyze the relationship between the type of rewards and the effect of the gamification rules. We note that rankings are often used by software engineering studies that focused on gamification. According to the survey by Pedreira et al. [6], ranking is the third most used reward in the software engineering field out of 14 rewards.

After the experiments, a few subjects wanted to know their ranking as soon as possible. If we show not only their scores but also their rankings during the experiments, the effect of the gamification rules could be enhanced. However, showing such real-time rankings in actual software development is not easy because each developer does not simultaneously deal with similar tasks (e.g., a developer could be making a source code, whereas another developer could be performing software testing at the moment). In contrast, showing such real-time rankings in programming classes is not difficult.

5. Conclusions

This study experimentally evaluated the effect of gamification rules on work results such as working time. As a result, we observed the following tendencies:

- For code review, when we applied a rule that made subjects aware of working time, it did not shorten the review time. However, it also did not reduce the number of found faults.
- For the coding tasks, working time may have been shortened when we applied a rule that made developers aware of working time.

Our future research work will evaluate the influence of

rules on the quality of coding, and to analyze the relationship.

Acknowledgments

This research was partially supported by the Japan Society for the Promotion of Science (JSPS) [Grants-in-Aid for Scientific Research (C) and (A) (No.16K00113 and No.17H00731)]

References

[1] S. Arai, K. Sakamoto, H. Washizaki, and Y. Fukazawa, "A Gamified Tool for Motivating Developers to Remove Warnings of Bug Pattern Tools," Proc. International Workshop on Empirical Software Engineering in Practice (IWESEP), pp.37–42, 2014.

[2] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining "gamification,"" Proc. International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek), pp.9–15, 2011.

[3] T. Ichinose and H. Uwano, "Evaluation of task performance with different entertainments in gamification," The transactions of Human Interface Society, vol.18, no.2, pp.65–76, 2016 (in Japanese).

[4] S. Khandelwal, S.K. Sripada, and Y.R. Reddy, "Impact of Gamification on Code review process: An Experimental Study," Proc. Innovations in Software Engineering Conference (ISEC), pp.122–126, 2017.

[5] S. Ohji and H. Uwano, "Effect of review guidance to efficiency of software review," IPSJ SIG Technical Reports, vol.2014-SE-185, no.2, pp.1–8, 2014 (in Japanese).

[6] O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering – A systematic mapping," Information and Software Technology, vol.57, no.1, pp.157–168, 2015.

[7] L.C. Stanculescu, A. Bozzon, R.-J. Sips, and G.-J. Houben, "Work and Play: An Experiment in Enterprise Gamification," Proc. ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW), pp.346–358, 2016.