

Can Gamification Encourage Debugger Usage?

Teresa Dreyer

PLTlab.de

Marburg University

tdreyer@mathematik.uni-marburg.de

Christoph Bockisch

PLTlab.de

Marburg University

bockisch@acm.org

Abstract

Computer Science students often don't make use of the debugger. If they do, they only use its basic functionality. We propose encouraging students to better exploit the debuggers' potential by providing them with debugging related gamification elements integrated into their usual IDE. Among several suitable gamification elements, we are focusing on achievements which are well-researched and work well across different contexts, including software engineering. We have developed an IntelliJ IDEA plugin that awards achievements for reaching specific milestones when using the debugger such as using the stepper for the first time or setting 100 breakpoints. We conducted an experiment with second semester Bachelors' students in which we required them to use the plugin while working on programming tasks. Half of the participants were shown the debugger-related achievements and the other half used a non-active version of the plugin where usage data was only collected but not displayed. The results from an initial analysis of students' self-report data look promising.

1. Introduction

Debugging makes up a significant part of software development, yet developers are rarely formally trained in debugging techniques (Ko, Bose, Chowdhury, Seyam, & Brown, 2023; Perscheid, Siegmund, Taeumel, & Hirschfeld, 2016). Two other observations are common and related: Computer Science students often struggle with debugging (Yang, Baird, O'Rourke, Brennan, & Schneider, 2024) and underuse the debugger (Ko et al., 2023). Often they don't use a dedicated debugger at all and instead debug using print statements (Ko et al., 2023). If they do use it, they often only use basic features like line breakpoints although the debuggers integrated into modern IDEs provide a substantially broader range of functionality including different breakpoint types as well as a dedicated debugging view with functionality for inspecting a suspending program frame by frame such as a stepper and an overview over current variable values and active threads.

Gamification, the practice of taking elements of games and applying them to non-game contexts, has successfully been used to teach people desired behavior in many areas. Different categories of gamification elements exist such as rewards, streaks, leaderboards, and progression systems. The inclusion of such elements in user facing applications is a popular means to increase users' motivation and persistence. Notable examples are for example streaks for daily activity in the language learning app Duolingo or the social media platform Snapchat, leaderboards and a level-based progression system also in Duolingo, and badges on the Q&A-website Stack Overflow. The application of gamification strategies to software engineering has been extensively studied for several years, both within and outside of a dedicated educational context with overall promising results.

We are investigating how the inclusion of debugging-related gamification elements into an IDE familiar to our students can help increase their motivation to use the IDE's debugger more as well as exploit more of its feature range. As a consequence of increased usage and feature exploitation, students are also expected to implicitly learn how to use the debugger as well as more effective debugging strategies.

2. State of the Art

Among the various available gamification elements, achievements in general have been stated to be one of the most basic and elementary ones (Dubois & Tamburrelli, 2013): The relationship between the user's behavior and reward is instantaneous, they do not rely on a multi-day process (in contrast to

streaks) or on other users' behavior (as would be the case with leader boards). Yet, they are generally effective at increasing motivation, which was also confirmed in the software engineering context (Souza, Constantino, Veadó, & Figueiredo, 2017; Porto, Jesus, Ferrari, & Fabbri, 2021) whereas the picture for competition based gamification elements like leaderboards is more mixed (Souza et al., 2017). These properties make achievements a very suitable starting candidate for a gamification element to study.

In general, gamification research in software engineering has shown very promising results: Souza et al. (2017) collected survey data indicating that gamification elements can be beneficial in software engineering education. In a recent systematic mapping study, Porto et al. (2021) were able to confirm this. Two studies from the area of software testing are particularly interesting in comparison with our endeavor: In an experiment spanning over a few weeks, Ren (2023) confirmed that students' test-driven development practices can be positively influenced using gamification. Straubinger and Fraser (2024) introduced IntelliGame, an IntelliJ plugin that rewards the user for desired testing behavior with achievements. Using a controlled experiment, they were able to show that students using their plugin displayed more desired testing behavior than students who did not use the plugin. Because of the generally promising results of applying gamification techniques to software engineering contexts, it is plausible to assume that similar effects should be discoverable when gamifying debugging in a similar manner.

3. A Plugin to Reward Debugger Use

As a platform for researching the influence of gamification elements on debugger usage, we have developed an IntelliJ IDEA plugin. The choice of the environment IntelliJ IDEA plugin is primarily motivated by our students' familiarity with it.

The plugin was designed with the explicit goal of conducting controlled experiments in mind. When conducting controlled experiments, it is central to eliminate potential systematic confounding variables. Comparing study participants working with the plugin to study participants with an unmodified version of the IDE would introduce at least two potential confounding variables: a) the general layout of the IDE would be different and b) students working with the plugin would likely need to install it first. Instead, we propose a different approach inspired by experimental designs in medical and clinical psychological research: We have implemented a "*placebo*" version and an "*active*" version of the plugin: Depending on a code the participant enters when setting up the plugin, they receive either the *placebo* version consisting of an otherwise empty sidebar with an export button for debugger usage data or the *active* version in which the implemented gamification elements are enabled. With general environment layout and installation differences eliminated as potential confounding variables, it is possible to use our plugin for conducting a controlled experiment in which the only systematically differentiating factor between participants is whether they have access to the gamification elements or not.

Due to their simplicity, achievements were selected as a first gamification element to examine. This decision is backed by the research by Dubois and Tamburrelli (2013) and Souza et al. (2017). To encourage students to explore different types of breakpoints as well as the debugger's stepper, 8 breakpoint achievements corresponding to the 8 breakpoint types available in IntelliJ IDEA were implemented as well as 5 achievements associated with using the stepper to explore the program state frame by frame. Additionally, 3 hidden achievements were implemented for which only question marks are displayed until they are unlocked. The hidden achievements aim at freely encouraging exploration of additional debugger functionality which is not covered by the other achievements. Exploring the program state and other more advanced features are not currently covered by our achievements due to time constraints.

Plugin Layout The *active* version of the plugin includes a sidebar that shows the available achievements, see figure 1 for an excerpt. Both the *active* and the *placebo* version of the plugin collect debugger usage data associated with the implemented achievements which can be exported by using an `export` button. In the *placebo* version, only this export button is displayed. The general layout and design of the *active* version of the plugin is inspired by IntelliGame (Straubinger & Fraser, 2024).

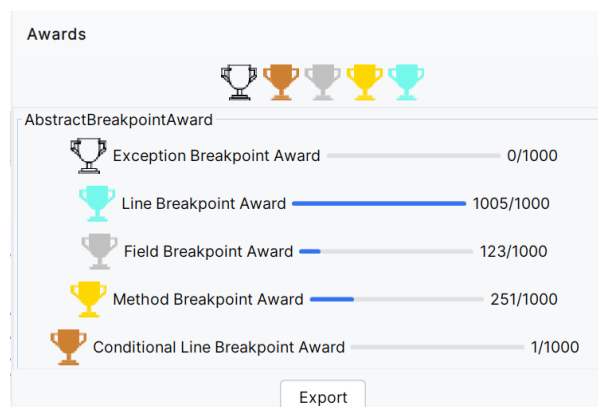


Figure 1 – Screenshot of the sidebar for the “active” version of our plugin showing part of the achievements and the export button. In the “placebo” version, only the export button is displayed.

Similar to IntelliGame, our implementation of the *active* plugin version also uses a level system for the breakpoint and stepper related achievements: For the first instance of an achievement related behavior, for example setting a line breakpoint, the user is awarded the bronze level award, for future instances silver, gold and platinum achievements are awarded with an increasingly higher distance between achievements, for the 100th, 250th and 1000th instance of an achievement-associated behavior, respectively. A progress bar shows how far the developer is from the platinum level. In addition, popups notify the user when an achievement is reached, and on occasion as well for additional milestones, with similarly decreasing frequency. The hidden achievements only have one level each.

4. User Study

We have evaluated our plugin in a university course that is mandatory for Bachelors’ students in Computer Science, Business Informatics and Data Science. The course in question is mainly attended at the end of the second semester of the Bachelor’s degree. At this point in time, the students have a basic orientation in object oriented programming, algorithms, data structures, and other programming concepts. All students were required to follow the provided tasks. After completing the tasks, they were allowed to freely choose whether they wanted to share their data for research purposes. We formulated the following general hypotheses:

- A) *Students using the active version of the plugin will have larger values than students using the placebo version for the questions in the self-report that are associated with debugger usage during the class and expectations for future debugger usage.*
- B) *Students using the active plugin version will use the debugger more than students with the placebo version.*

4.1. Experimental Setup

All participants were instructed to install the plugin together with a compatible version of IntelliJ IDEA. Afterwards they were given a code to enable the plugin. The students then worked on provided programming tasks and were instructed to use the debugger. Most of the students worked in teams of two. The students were divided into two experimental groups: The treatment group was provided with the *active* version of our plugin whereas the control group was provided with the *placebo* version. For both groups, debugger usage data was collected and exported afterwards. Students working in pairs were assigned to the same experimental group. We collected self-report data in the form of a questionnaire from 111 students of which 91 students agreed to have their data used for research purposes. Two of them were excluded because it could not be reconstructed in which experimental group they had been in.

4.2. A Look at the Data

In an initial analysis, we focused on hypothesis A) and considered only the self-report data, particularly the students' responses to the four relevant questions. For each one, we performed a simple linear regression with the experimental condition (control vs. treatment condition) as predictor variable and the score in the respective questionnaire item as outcome variable. The predictor variable was dummy encoded with the control condition encoded with 0 and the treatment condition encoded with 1. Additionally, we performed a simple linear regression using the same predictor variable and the sum score of the four questionnaire items as outcome variable. The results can be found in table 1. The exact wording of the corresponding questions is found in table 2 in the appendix. While regression weights are rather small and not statistically significant, for all relevant questions from the questionnaire as well as the sum score, regression weights are positive.

Table 1 – Results of the initial data analysis

Dependent Variable (DV)	intercept	regression weight	intercept (DV standardized)	regression weight (DV standardized)
DV1: Intensity of use during class (1 to 5)	1.97	0.045	−0.026	0.049
DV2: Frequency of use after class (1 to 5)	3	0.063	−0.03	0.056
DV3: Better use of features after class (1 to 5)	3.048	0.186	−0.087	0.164
DV4: Influence of plugin on likelihood of using the debugger (1 to 5)	3.095	0.16	−0.158	0.3
Sum score over all 4 relevant questions of the questionnaire (4 to 16)	11.119	0.455	−0.09	0.171

4.3. Interpretation

With the regression weights not being statistically significant, interpretation is limited. Additionally, the hierarchical nature of the data caused by the students working in teams was not taken into account here, further limiting conclusions. Taking these limitations into account, we consider the initial results as tentatively positive for a first step into the direction of our envisioned research: All regression weights being positive is in accordance with our expectations.

5. Conclusion and Future Work

We have introduced a tool that enables researching the effect of gamification elements on debugging behavior in the form of an IntelliJ plugin set up for conducting controlled experiments. We have performed a first experiment for which the analyzed self-report data looks tentatively positive.

With our research into the matter just at the beginning, we consider the initial results encouraging and intend in a next step to take a more in-depth look at our existing data, starting with the evaluation of the same students' objective debugger usage data.

Looking back at the performed experiment, we have observed several things that we would like to improve for future experiments: Letting the students work in teams was counter productive due to introducing a hierarchical element into the data. While no systematic confounding variables are expected to have been introduced this way, proper statistical procedure requires taking the hierarchical nature of the data into account. Such methods, however, typically rely on data sets being substantially larger than ours. For future experiments, having the students work individually would elegantly solve the problem. Additionally, the experiment took course over one day only. We did not take the short timeframe into account when setting the limits for the breakpoint- and stepper-associated achievements: Reaching the gold level for the line breakpoint achievement for example would have required a student to set 250 breakpoints which is not a realistically achievable amount in one work day. This may have artificially

limited the motivating factor of the introduced gamification element. Future research should take this into account and set appropriate limits.

Regarding our tool, we are envisioning several extensions in future work: In the short term, we intend to implement additional achievements to cover more of the functionality of IntelliJ IDEA's debugger. As a medium to long term goal, we intend to extend our platform to include additional gamification elements. We are considering streaks, a point system as well as missions as possible next research avenues. The combination of several gamification elements can also be evaluated. The platform is also extensible for other learning goals derived from software engineering education, for example refactoring and testing. While testing by itself is already covered in the research by Straubinger and Fraser (2024), it may still be interesting to research in combination with other software engineering education goals. Finally, the concept of a *placebo* version of a plugin or a different application to be tested in an experimental design can and should be transferred to other research areas within software engineering research.

6. Acknowledgments

Finn Brecher, Daniel Hinkelmann, Moritz Ole Krenzer and Davide Piacenza were involved in the implementation of the plugin.

7. References

- Dubois, D. J., & Tamburrelli, G. (2013, August). Understanding gamification mechanisms for software development. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering* (p. 659–662). ACM. doi: 10.1145/2491411.2494589
- Ko, M., Bose, D. B., Chowdhury, H. A., Seyam, M., & Brown, C. (2023). Exploring the barriers and factors that influence debugger usage for students. In *2023 ieee symposium on visual languages and human-centric computing (vl/hcc)* (p. 168-172). doi: 10.1109/VL-HCC57772.2023.00027
- Perscheid, M., Siegmund, B., Taeumel, M., & Hirschfeld, R. (2016, January). Studying the advancement in debugging practice of professional software developers. *Software Quality Journal*, 25(1), 83–110. doi: 10.1007/s11219-015-9294-2
- Porto, D. d. P., Jesus, G. M. d., Ferrari, F. C., & Fabbri, S. C. P. F. (2021, March). Initiatives and challenges of using gamification in software engineering: A systematic mapping. *Journal of Systems and Software*, 173, 110870. doi: 10.1016/j.jss.2020.110870
- Ren, W. (2023, December). Gamification in test-driven development practice. In *Proceedings of the 2nd international workshop on gamification in software development, verification, and validation* (p. 38–46). ACM. doi: 10.1145/3617553.3617889
- Souza, M. R. d. A., Constantino, K. F., Veadó, L. F., & Figueiredo, E. M. L. (2017, November). Gamification in software engineering education: An empirical study. In *2017 ieee 30th conference on software engineering education and training (csee&t)* (p. 276–284). IEEE. doi: 10.1109/cseet.2017.51
- Straubinger, P., & Fraser, G. (2024). Improving testing behavior by gamifying intellij. In *Proceedings of the ieee/acm 46th international conference on software engineering*. New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3597503.3623339
- Yang, S., Baird, M., O'Rourke, E., Brennan, K., & Schneider, B. (2024, November). Decoding debugging instruction: A systematic literature review of debugging interventions. *ACM Transactions on Computing Education*, 24(4), 1–44. doi: 10.1145/3690652

Appendix

Table 2 – Evaluated Questions from the Questionnaire

DV1: Intensity of use during class	Corresponding Question with Scale (from German Questionnaire)
Original Question (German)	Nach Ihrer eigenen Einschätzung: Wie intensiv haben Sie den Debugger während dieses Programmierpraktikumsblocks eingesetzt?
Original Scale (German) with assigned values	sehr wenig (1), eher wenig (2), neutral (3), eher viel (4), sehr viel (5)
Translated Question	In your own estimation, how intensely did you use the debugger during this block of the class?
Translated Scale with assigned values	very little (1), rather little (2), neutral (3), rather much (4), very much (5)
DV2: Frequency of use after class	Corresponding Question with Scale (from German Questionnaire)
Original Question (German)	Nach dieser Praktikumseinheit werde ich den Debugger häufiger verwenden als vorher.
Original Scale (German) with assigned values	stimme gar nicht zu (1), stimme nicht zu (2), neutral (3), stimme zu (4), stimme voll und ganz zu (5)
Translated Question	After the class, I am going to use the debugger more frequently than before.
Translated Scale with assigned values	completely disagree (1), disagree (2), neutral (3), agree (4), completely agree (5)
DV3: Better use of features after class	Corresponding Question with Scale (from German Questionnaire)
Original Question (German)	Nach dieser Praktikumseinheit werde ich die Möglichkeiten des Debuggers besser ausnutzen als vorher.
Original Scale (German) with assigned values	stimme gar nicht zu (1), stimme nicht zu (2), neutral (3), stimme zu (4), stimme voll und ganz zu (5)
Translated Question	After this class, I'll make better use of the debugger's features than before.
Translated Scale with assigned values	completely disagree (1), disagree (2), neutral (3), agree (4), completely agree (5)
DV4: Influence of plugin on likelihood of using the debugger	Corresponding Question with Scale (from German Questionnaire)
Original Question (German)	Wie verändert das Plugin, das Sie am Anfang des ersten Tages des Blocks installieren sollten, für Sie die Wahrscheinlichkeit den Debugger zu benutzen?
Original Scale (German) with assigned values	Es verringert stark die Wahrscheinlichkeit. (1), Es verringert eher die Wahrscheinlichkeit. (2), neutral (3), Es erhöht eher die Wahrscheinlichkeit. (4), Es erhöht stark die Wahrscheinlichkeit.
Translated Question	How does the plugin you were asked to install [...] affect your likelihood of using the debugger?
Translated Scale with assigned values	It greatly reduces the likelihood. (1), It rather reduces the likelihood (2), neutral (3), It rather increases the likelihood. (4), It greatly increases the likelihood. (5)