Maintaining Motivation in a Software Testing Team:

Mark Kardash- Senior QA Consultant

Written for the Association for Software Testing

13 March 2021

The purpose of this report is to recommend solutions for the lack of motivation in Software Quality Assurance teams. According to research, Software QA Testers can often lose focus on their tasks due to the work becoming mundane. This report contains an overview of the problem, its background, previously attempted solutions, a description and comparison of potential future solutions, as well as a final recommendation for how this issue could be dealt with.

According to a May 2016 study, many countries, such as India, Britain and Australia, are having difficulties finding people who are willing to work as software testers. The study has found that “this is becoming a key challenge, and that there is an increased need for testing-related training” (Deak, [Stålhane](https://www.sciencedirect.com/science/article/pii/S0950584916000045" \l "!), [Sindre](https://www.sciencedirect.com/science/article/pii/S0950584916000045" \l "!)). What’s worse, with no passion and drive, the working pace might slow down, lower quality bug reports will be created, less software bugs will be found, and critical bugs may be missed (Deak, [Stålhane](https://www.sciencedirect.com/science/article/pii/S0950584916000045#!), [Sindre](https://www.sciencedirect.com/science/article/pii/S0950584916000045#!)). Also, when uninterested, testers do their work in extreme hurry, striving to be done with it as soon as possible.

As you most likely already know, leaving a critical bug unattended can have severe consequences, from the need to recall the product to even loss of life. In 2015, for example, a software error in F-35 fighter jets caused the aircraft to fail to determine the number of targets they are detecting. Even though this particular event did not lead to any fatalities, no one can guarantee that something similar couldn’t happen again, with more calamitous results. Another example is that of a January 2018 warning about an inbound nuclear missile strike across the entire state of Hawaii, mistakeably sent out due to a software error in Hawaii’s Emergency Management Agency’s public alert system (Tricentis.com). While the warning was fortunately a mistake, this incident shows how a software error could spark mass hysteria.

Even though there is no way to link the above incidents to boredom or dissatisfaction of testers, these factors should not be excluded. And neither can software testing be called a “boring job”. Niranjan Limbachiya, includes a quote by Michael Bolton in a Linkedin article: “**Testing is continuous learning process by exploring, discovering and investigating the information you have”.** (Limbachiya, 1). The big truth is, however, that like many other positions, it involves certain repetitive tasks, which seem like a monotonous loop. This problem is, of course, recognized by many test team leads, and solutions to it have already been tried. It has been proven that the automation of testing tasks is not the ultimate solution, as it would take a great amount of time for it to be fully implemented. Neither will the exchange of testers around the globe prove effective, as, due to the global shortage of QA personnel, the field might be thrown off balance. (Deak, [Stålhane](https://www.sciencedirect.com/science/article/pii/S0950584916000045#!), [Sindre](https://www.sciencedirect.com/science/article/pii/S0950584916000045#!))

Having gone through the problem of “testing boredom” myself, I have conducted research and found several potentially effective solutions to the issue. After reading your mission statement and guiding principles, I believe that as a company which sees software QA as “a cognitively complex activity that requires critical thinking, effective communication, and rapid self-directed learning.”, you will be benefitted by my suggestions below.

The first solution that I would propose is diversifying testers’ tasks with games, preferably ones that help build team spirit. While some test team leads do hold “find the most bugs” contests, these games can take a different approach, focusing more on the quality, rather the quantity, of the bugs found. To make sure I am being understood, I will analyze this solution below.

Suppose an application, or a series of applications, needs to be tested by a team. This requires a lot of sitting and clicking on buttons, which, for some testers, might seem pretty routine. They might, however, reconsider this once a more specific goal enters the scene. For example, a team lead can organize a contest, in which a prize is to be given to the tester who finds the largest amount of critical bugs in a given period of time. The time period itself would mostly depend on the length of each particular project. Here is how it could go: At the beginning of a project, the team lead takes about ten minutes to announce and explain the contest to the testers. After this, the project begins. When a tester finds a critical bug, they file their test case, thus reporting it to the lead. The lead then puts a pin next to that tester’s name, either on a clipboard or electronic spreadsheet. By the time the project ends, the pins are counted up, and the tester with the most of them gets awarded the prize. At the end of the project, the team lead can take five or ten minutes to congratulate the winner before proceeding to the typical workday. The final prize can be a gift card to a store, or tickets to a highly anticipated film.

When games like this are presented to them, testers will actually have a much more personalized motivation to find high-quality critical bugs, rather than simply having to complete the testing before the release deadline. It would also create a sense of much needed challenge and competition, without necessarily sparking serious rivalry within the team.

Even though, as described here, the solution may have its flaws, there are ways these flaws can be taken care of. For example, one would be concerned that in their efforts to find critical bugs, testers would disregard smaller issues when coming across them. However, this could be fixed by arranging a bug hunt for smaller issues after the critical bugs are done for. Uplifted by the contest, testers will not be as bored to look for smaller bugs as they were before.

Holding a contest like this also ensures that the critical bugs will be resolved first, giving the chance of wrapping up the testing completely before the time comes to release the product. You might also express the concern that it might be too costly to organize such contests all the time. I believe, however, that this could be solved by creating a specific schedule for the contests, and spending a “healthy” budget on designated prizes.

A second solution I would propose is the proper distribution of work among testers. Chances are that some testers are bored of their work simply because they either focus on only one task for too long, or because the process seems like a lot of work to be done. When dealing with such a situation, a good way to motivate testers is to reassign tasks. For example, if a team member is exhausted after several weeks of looking for bugs, team leads could assign him to work on documentation instead, getting the person previously in charge of documentation to do testing, and vice versa. When first meeting their team, leads can ask each member to complete a short survey, asking them about how long they can perform a repetitive task for. Then, based on the average of the responses, the lead can decide a timeframe on which the reassignment could be happening, to fit with the release of each project. This approach has the chance to both add dimension to a software tester’s work, and improve their skills in multiple tasks. It would function in a similar way to when a tester takes a day off, and another needs to replace them. Team leads would also not need to worry about the effectiveness of this exchange, as most testers are equally well-trained in both the technical and documentation aspects of QA, and have to g through both when taking courses.

In order to decide which of the two solutions is more effective and feasible, I have decided to analyze and compare them based on certain points. These points include time to implement, feasibility within the workplace, potential cost, and chances of success (i.e. level of effectiveness).

SOLUTION 1: BUG CONTEST

The timeframe for organizing such a contest from initial idea and purchase of materials could take about a week. Once implemented, however, it will be much easier, since all the team lead would have to do is announce it prior to start if a project.

The solution appears very feasible as well. Because projects are an everyday part of Software QA, holding a contest or competition will not necessarily directly interfere with the workflow. On the contrary, if successful, the contest has the chance of motivating testers to complete their work fast, while at the same time paying attention to the quality of the issues they find.

The bug contest, however, does have some financial costs. The cost of a pack of 100 “Push Pins” on the website [Push Pins | JAM Paper](https://www.jampaper.com/ecom/index.asp/show/detail/cat/59/prd/224/c/20954/s/990?msclkid=bcfd3456467b1eec0d6b54076a794866&utm_source=bing&utm_medium=cpc&utm_campaign=(ROI)%20Office%20Helpers%20-%20Shop%20%5BNEW%5D&utm_term=1100400348667&utm_content=Pushpins%20%5BNEW%5D) is $7.95. Can be purchased from any paper store for approximately the same price. (The company may have enough pins of its own, so purchasing may not be necessary). As for gift cards and other prizes, this is a bit tricky, as an entire separate budget might need to be designated for purchasing the gifts. In order to make this easier, the team lead must set a clear limit on how expensive a prize should be. I would recommend keeping each price under $50.

This solution has a great chance of succeeding, provided that the budget and materials for it are accounted for correctly. Contests and competitions are often applied in many other fields as well. It is also in human nature to face and respond to challenges, so the chances that testers will be motivated by such a technique are quite high.

SOLUTION 2: PROPER WORK DISTRIBUTION:

Although there is no specific timeframe for this solution, I would suggest two weeks of experimental training for testers to get used to it. After this, all that will need to happen is for the testers to get notified at least a day prior to the switch.

In terms of feasibility, this idea might be slightly problematic, as the change of task of a group from testing to documentation might be confusing and somewhat time consuming. It would always need to be remembered when each group of testers performs each task. Relocations to different corers of the office might also be needed, which could be both time consuming and annoying.

No materials need to be purchased for this solution, so the financial cost is 0.

When it comes to success, this solution is a little bit more ambiguous. We cannot know for sure that a simple change of tasks will motivate everyone, since some testers do not plan for QA to be their permanent job, and their boredom is not dependent on the tasks they are doing.

After analyzing both of my proposed solutions based on multiple factors, I would like to suggest that you consider implementing a bug contest. Although it does require extra costs, and some extra considerations to be successful, it has greater chances of motivating testers than the switching of tasks. At the end, it all comes down to appreciation, rather than the prize itself. Humans love to be appreciated, and testers love when their work is appreciated. I must therefore conclude this report by saying that I highly recommend bug contests to be used in the QA field, and truly hope that your association will spread my recommendation across the field.

Works Cited:

Limbachiya, Niranjan. “Is Software Testing a Boring Job?” LinkedIn, March 30, 2017. https://www.linkedin.com/pulse/software-testing-boring-job-niranjan-limbachiya/.

Deak, Anca, Tor Stålhane, and Guttorm Sindre. 2016. “Challenges and Strategies for Motivating Software Testing Personnel.” *Information and Software Technology* 73: 1–15.

Tricentis Team. 2018. “Real Life Examples of Software Development Failures.” Tricentis.Com. December 5, 2018. <https://www.tricentis.com/blog/real-life-examples-of-software-development-failures/>.

Fanadka, Karim. 2015. “10 Best Practices to Help QA Teams Deliver Quality Enterprise Software.” Techbeacon.Com. TechBeacon. May 28, 2015. <https://techbeacon.com/app-dev-testing/10-best-practices-qa-teams-deliver-quality-software-fast>.

Hossellman, Emily. n.d. “How to Incentivize Software and Web Site Beta Testers.” Centercode.Com. Accessed March 16, 2021. <https://www.centercode.com/blog/2014/11/how-to-incentivize-software-and-web-site-beta-testers>.

“For Professional Software Testers - Association for Software Testing.” 2010. Associationforsoftwaretesting.Org. September 6, 2010. <https://associationforsoftwaretesting.org/>.

Francino, Yvette. 2015. “13 Ways to Motivate Software Engineers and Dev Teams.” Techbeacon.Com. TechBeacon. August 7, 2015. https://techbeacon.com/app-dev-testing/13-ways-motivate-software-engineers.

Bakos, Jennifer. n.d. “6 Good Beta Test Incentives Options (and 3 to Avoid).” Centercode.Com. Accessed March 17, 2021. https://www.centercode.com/blog/2015/06/6-good-beta-test-incentives-options-and-3-to-avoid.

“Push Pins.” n.d. Jampaper.Com. Accessed March 17, 2021. https://www.jampaper.com/ecom/index.asp/show/detail/cat/59/prd/224/c/20954/s/990?msclkid=bcfd3456467b1eec0d6b54076a794866&utm\_source=bing&utm\_medium=cpc&utm\_campaign=(ROI)%20Office%20Helpers%20-%20Shop%20%5BNEW%5D&utm\_term=1100400348667&utm\_content=Pushpins%20%5BNEW%5D.