**Faye Van Roekel**

**February 9, 2020**

**Web-430**

**Code Reviews at Google**

Google is a technology company that employs thousands of developers. In 2010, Google had their development engineers working off a single code source they called the trunk. The engineers were performing over 5,500 commits to the trunk per week. The engineers were producing 20+ changes to the trunk every minute, which was 50% of their code base being changed every month (Kim, Humble, Debois, & Willis, 2016).  Managing and preventing issues from happening becomes an enormous task with the quantity to changes made to the trunk.

Google ended up needing to increase more lead time to get the changes made. Typically, the larger the change submitted for code review, the larger the lead time required to sign off on the change. Lead time is the latency between the initiation and execution of a process. The size of the changes that needed to be reviewed was increasing as well. This made code reviewers focus on the code’s style for readability and language, made code more transparent across teams, and created consistency for sub-trees (Kim, Humble, Debois, & Willis, 2016). Google implemented that all team members send their commits through mandatory code reviews that cover code readability for languages, ownership assignments for code sub-trees to maintain consistency and correctness, and code transparency and code contributions across teams.

Randy Shoup, engineering director at Google, started a project to solve a technical problem that Google was experiencing. The project took months to write and had over 3,000 lines of code within it. After the project was completed, he requested for a peer to review his work. It took the peer days of work to complete the review and the peer requested to not review an entire project at one time. Randy Shoup discovered that working code reviews into a daily routine would streamline the process and makes it easier to find issues.

Google moved to a code review process where it was incorporated into their daily work. Google also felt that code review would be able to make more than one developer familiar with each piece of code, and that it would increase the knowledge base within the company.

Best practices for code review include reviewing less than 400 lines of code at a time. The brain can only process so much information at a time. SmartBear on Cisco Systems performed a study and in programming found that when a developer tries to review more than 400 lines of code, the ability to find error, bugs, and defects were diminished (SmartBear, n.d.).

Additionally, a team should set certain metric when reviewing. Before beginning, the team needs to decide how it will measure the effectiveness of peer review as well as name a few quantifiable goals. A process should also be established for fixing the bugs that are found.

Finally, a positive code review culture can help encourage others. Managers should create a culture of collaboration, teamwork, and learning in their work centers. By doing this it will also help foster better coding. Peer review can help nurture cleaner commits, the sharing of knowledge, consistency and legibility in code writing, prevent accidental errors, and meet compliance demands.

**References:**

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2017). *The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations*. Portland, OR: IT Revolution Press, LLC.

SmartBear, (n.d.), Best Practices for Code Review, Retrieved: February 9, 2020, from: <https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/>

O'Neill, T. (2017, June 19). Code Health: Too Many Comments on Your Code Reviews? Retrieved February 9, 2020, from <https://testing.googleblog.com/2017/06/code-health-too-many-comments-on-your.html>