Code reviews can offer tons of benefits before submitting code for testing. Even large companies such as Google reviews their peers code before submitting. Google in 2016 had over 2,500 commits of code each week that needed to be reviewed and chances are they still review the code before submitting to tests. But there are several things to consider when reviewing code such as length of code and time, checklists, code annotations and promoting a culture of positivity for reviewing.

The first things to consider for code reviews would be the amount of time and length of code for reviewing. It is recommended that you review 400 lines of code or less. Code reviews should also be about sixty to ninety minutes long and no more than that. (*Best Practices for Code Review*, n.d.) You should be able to get through 500 lines per hour. (*Best Practices for Code Review*, n.d.) According to smartbear.com “a review of 200-400 lines of code over sixty to ninety minutes should yield 70-90% defect discovery.” (*Best Practices for Code Review*, n.d.) Its also recommended to take our time reviewing code and to not spend more than an hour at a time, otherwise this could affect the quality of our work. (*Best Practices for Code Review*, n.d.)

It’s also important to set the goals of the review and capture certain metrics as well. (*Best Practices for Code Review*, n.d.) Useful metrics to consider tracking are the inspection rate, which is the speed the review is done at. (*Best Practices for Code Review*, n.d.) Another metric to consider would be the defect rate which is how many errors are found each hour of review. (*Best Practices for Code Review*, n.d.) And another suggested metric to track is the density of defect which is how many errors are found in each line of code. (*Best Practices for Code Review*, n.d.) Back these metrics up with a checklist to further progress on your goals. The checklists could help your team not make the same mistakes or errors and will “provide team members with clear expectations for each type of review” (*Best Practices for Code Review*, n.d.)

Another thing to consider with effective code reviews is annotating your code before you send it off for review. It would be extremely helpful if the other software developers knew exactly what each functions intent was. The result would be that more bugs will be found. (*Best Practices for Code Review*, n.d.) And often according to smartbear.com the developer will find errors in their code as they are writing their code attribution. (*Best Practices for Code Review*, n.d.)

Lastly something to consider about code reviews is establishing a positive culture around it. This is an effort to promote learning within the team. (*Best Practices for Code Review*, n.d.) Each bug should provide the team with an opportunity to learn. It also allows for the more senior developers to mentor the less experienced developers. (*Best Practices for Code Review*, n.d.)

There are many factors in improving code reviews. But the biggest changes to be made is probably the amount of time and lines of code to be reviewed and supporting a positive culture of learning during the code reviews. These two items are very beneficial. Where as one will help errors be found at a better rate than trying to read over thousands of lines of code. The other will help the team learn and succeed which in the long run will provide the best possibly product.

References:

*Best Practices for Code Review*. (n.d.). smartbear.com. Retrieved June 29, 2023, from <https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/>

Kim, G., Debois, P., Willis, J. O., & Humble, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. https://dl.acm.org/citation.cfm?id=3044729