Feedback Culture and Code Review

by Vlad Ungureanu

LEARN stuff IO

Agenda

Feedback Culture
and
Code Review

- What is Feedback ?
- Feedback Culture
- How to give Feedback ?
- Positive Feedback
- Negative Feedback
- Handling Negative Feedback
- Code Review
- Why Review Code ?
- How to Review Code ?


I need someone that laughs at all my jokes.
You know, honest feedback.

What is feedback ?

- Crucial mechanism for reaching any goal.
- Allows you to adjust to the objective reality
- Control structure to avoid own perception or partially real assumptions.
- Self evaluation of own aptitudes.
- Understanding of the actual situation and the action that lead to this step.
- Concrete understanding of future, available actions.

Feedback culture

- Knowing and understanding individual needs
- Discovering available and sustainable resources
- Establishing a common language, difficult to interpret
- Avoiding prejudice and sophisms
- Efficient planning methodology
- Feedback as part of everyday life

How to give Feedback ?

- State it clear and concise
- Describe the situation, not the person
- Provide rational arguments
- Put focus on actions, not on introspection
- Limit feedback to available resources
- It should not express a necessity (avoid the words “need” or “must”)
- Ask for the other persons feedback
- It offers at least one practical way to solve the problem


LET ME BE CLEAR
I AM ONLY LOOKING FOR POSITIVE FEEDBACK
quickmeme.com

Positive feedback

- Focused on the positive aspects
- Express optimistically and comfortable
- Easy to receive
- Easy to provide
- Might make use of ambiguity
- Specific to parents, tutors
- Used to encourage and reinforce a specific behavior

Negative feedback

- Focused on things that could to be improved
- Express realistically
- Difficult to receive
- Difficult to give
- Specific to mentors, leaders
- Used to solve a specific problem

Handling Negative Feedback

- Evaluate received feedback, as the person giving us the feedback might have made a mistake
- If you are sure the feedback is not relevant give feedback to the other person
- If the feedback is relevant we try to associate it to the problem that caused the negative feedback
- Asks questions in order to clarify things and try to appear interested and reliable.
- Setup concrete actions and objectives
- Use negative feedback to understand the other person (needs thoughts, requirements)
- Understand that all forms of feedback are actually a form of interest for personal development and the company


EVERY TIME YOU
SKIP CODE REVIEW
A KITTY DIES
imgflip.com

Code Review

- Catch bugs
- Ensure code is readable and maintainable
- Make sure the implementation respects standards and clean code
- Make sure the system is developed according to the system design
- Spread knowledge of the code base throughout the team
- Get new people up to speed with the ways of working
- Expose everyone to different approaches

Why Review Code ?

- *"Any stupid can write the program that computer understands but only good programmers write code that humans understand"* – Martin Fowler

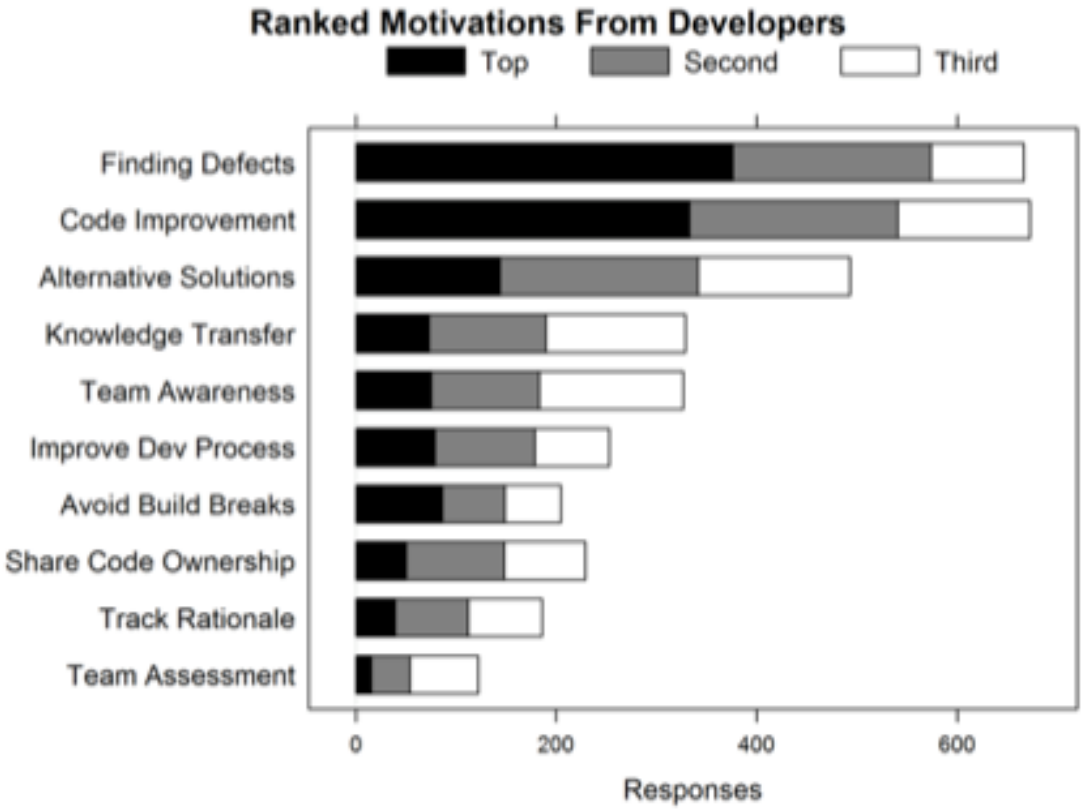
- *"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."* – Bill Gates

- *"Novices insert corrective code; experts remove defective code."* – Richard Pattis

- "Code review is like a pair of pants. If you work at home, pants are optional. But if you are in public, you'd better make sure you have pants on, even if no one looks at you." – Anonymous

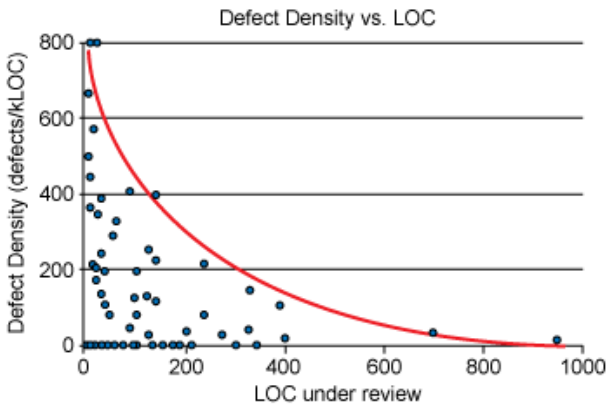
- "Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." – Martin Golding

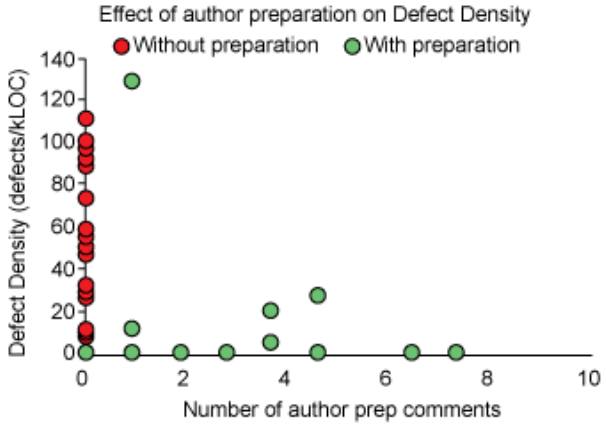
Why Review Code ?



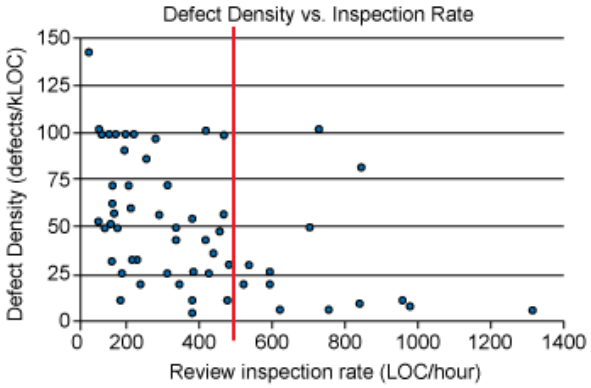
How to Review Code ?

- *Maximum 200~400 lines of code*





- *Spend maximum 1 hour*



- *Only signed classes*

How to Review Code ?

- Review all committed code
- Write your findings in a file, preferable after an common template
- Make sure it is readable, standardized and functional
- Make sure it does what it is supposed to do
- Make sure it does everything it is supposed to do
- Make sure it is aligned with current design requirements
- Code problems and suggest improvements
- Always personally communicate findings of code review
- Select actions and follow up on their implementation
- Use code quality metrics to guide your review

What to look for: Naming

- The name should accurately describe what the thing does.
- Do not use shortenings, only use well understood abbreviations.
- If the name looks awkward, the code is probably awkward.
- Proper exception management
- Document confusing sections of code

LEARN stuff IO

What to look for: Classes

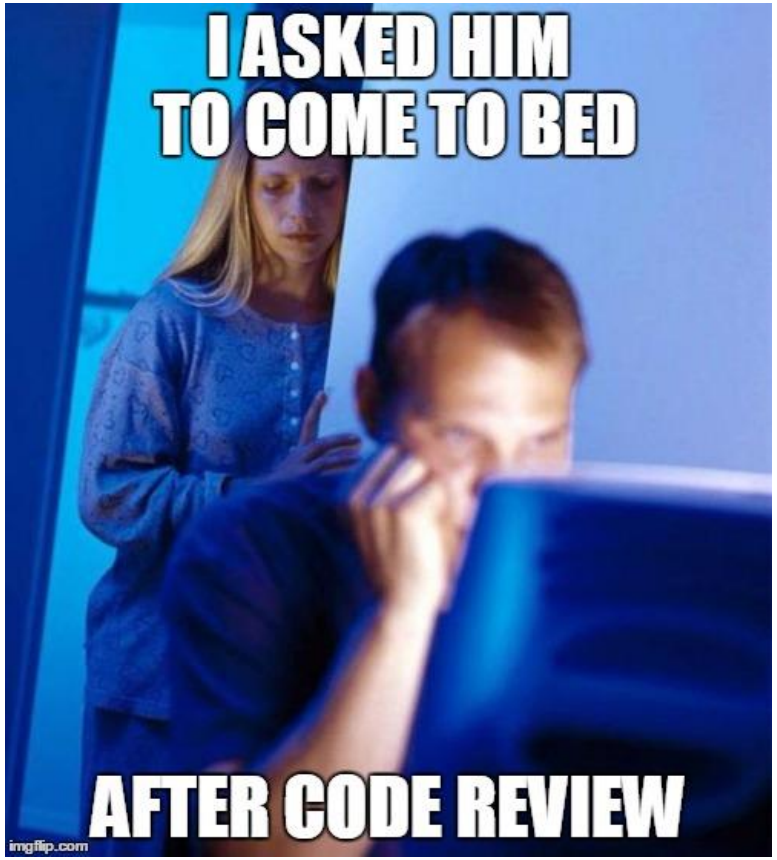
- The name should accurately describe what the class does.
- Classes typically represent data or services, be clear which your class is.
- Design your object oriented schema deliberately.
- A class should be small.
- A class should have one responsibility only.
- A class should have a clear contract.
- A class should be decoupled from its dependencies.
- Favor composition over inheritance.
- Avoid static classes and methods.
- Make the class immutable if possible.

What to look for: Interfaces

- Rely on interfaces rather than concrete classes wherever possible.
- An interface is a contract for interaction.
- An interface should have a single purpose.
- All methods in an Interface have proper comments.

What to look for: Principles

- Single responsibility principle: a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)
- Open/closed principle: classes should be open for extension, but closed for modification."
- Liskov substitution principle: objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program (design by contract).
- Interface segregation principle: many client-specific interfaces are better than one general-purpose interface.
- Dependency inversion principle: depend upon Abstractions, do not depend upon concretions.
- YAGNI – You aren't going to need it: never over-engineer, never add code that you consider you might need in the future, never try to optimize performance unless performance is an issue and never for design patterns if it is not required.


I ASKED HIM
TO COME TO BED
AFTER CODE REVIEW
imgflip.com

THANK YOU!

Vlad Costel Ungureanu
ungureanu_vlad_costel@yahoo.com

This is a free course from LearnStuff.io
– not for commercial use –