SEMINAR PRESENTATION - CODE REVIEWS

Stelian Adrian Stanci - UO277653

Martin Beltran Díaz - UO276244

Joaquín Salustiano Britos Morale - UO276044

0 - WHAT IS A CODE REVIEW?

When it comes to writing code, we always tend to write on-the-go code. Preparing a change for review is not only a way of preventing us from writing odd and difficult to read programs, but also a good practice so that future developers that read our code find it readable and cohesive with the rest of the application.

Code reviewing also allows you to get feedback in a timely manner, turning you into a better developer.

1 - CHALLENGES

1.1 - Challenges that can appear for code authors

Code authors can find a series of problems in the process of code reviewing, which can go from even before the code review has begun all the way to the results. In this section three of the most common challenges are discussed.

- The author cannot find any reviewer who is suitable for the review, or who is willing to review the code. There can be difficulties in knowing who to ask.
- The code reviewer takes too much time to give feedback, which can slow down the development speed of the coder.
- The feedback is not very useful, the reviewer focuses on small things (such as the names of the variables) instead of other important ones (for example the patterns used).

There are also other challenges that are worth mentioning, such as how to write the documentation for the changes in order for them to be as clear as possible to the reviewer, or sometimes the reviewer rejects the changes but does not state the reasons clearly.

Other challenges come from the means of communication, for example, when doing the review on the phone, after the call ends you probably have forgotten some things that were talked about, and also having several different communication channels can be confusing and slow down the developer.

1.2 - **Challenges that can appear for code reviewers**

The reviewers also have to cope with some problems, with the most recurrent ones being:

- Reviewing code takes too long. Understanding the code and changes made can be a long process, especially if the project is big and you have to review many.
- Other challenges that have to do more with the career of the reviewer, such as their job as reviewers not being very appreciated or taken into account in job applications. Also many reported not having been trained.

As we can see there are many problems that can appear when code reviewing, and in order to prevent or solve them there are a series of best practices that can be applied.

2 - BEST PRACTICES

2.1 - **Best practices for code authors**

-> Aim for small and incremental changes is basic. Sending clusters of related small challenges to reviewers, will help them understand the overall nature of the changes.

-> Is important to know when to do a review, but even more important when to skip it. Non-logical changes or stylistic fixes, for example, rarely need to be reviewed.

-> Select the reviewers that fit the best. Maybe selecting them by expertise, by field experience, or allowing them to volunteer.

-> Before committing any final change, document every decision.

2.2 - **Best practices for code reviewers**

-> Don’t underestimate a small change. Devoting the right amount of time for reviewing a change is key, even a single line of code can create an “avalanche effect”.

-> Frequent reviews with less changes is better than less reviews with a great amount of requests. But remember to not overwhelm the reviewer.

-> Review first core issues.

-> Communicate feedback by channels that record and save that feedback for the future. Maybe someone can find it useful.

2.3 - What about organizations?

The main function of the organization is, mainly, ensuring a positive review culture, so that every feedback or discussion is constructive, and providing the appropriate tools and mechanisms. Everyone should be humble, trustable and respectful.

3 - EVERYTHING IN SOFTWARE ARCHITECTURE IS A TRADE OFF

As you have already seen in theory classes everything in software architecture is a trade-off, and this is no less for best practices in code review. Best practices trade offs are infinite but there are some of them which are like patterns and can be seen in many different places, for example:

- **Time constraints**

  Whenever the code review is a reason for a bottleneck in the development of the project.

- **Rigid policies**

  Whenever developers must fulfill tasks before delivering the work, so they bundle it and send it to the reviewer, resulting in a poorly done and less coherent review.

- **Reviewer selection**

  Choosing the right person to review the code is important since we are depriving them from doing other stuff like bug fixing or coding.

REFERENCES

https://www.michaelagreiler.com/respectful-constructive-code-review-feedback/
https://www.michaelagreiler.com/wp-content/uploads/2019/03/Code-Reviewing-in-the-Trenches-Understanding-Challenges-Best-Practices-and-Tool-Needs.pdf