SDEVEN Software Development & Engineering Methodology

Version: 7.0.6
Release date: 230626

Code Review (SDEVEN.68-COREV)

Table of Content

Preamble and objectives

This section is about code review. The main **objectives** of a code review are:

- Sharing knowledge
- Sharing responsibility
- Improving code structure
- Learning

A good and effective code review will cover all those aspects.

Technical objectives and the process

- The main purpose is to "detect" those code parts that can be *generalized and reused*
- Another purpose is to check the conformity to appropriate standards and practices (for example for a `Python` code to check if it respects `PEP` indications, or for a web server front end application if `HTML` specifications was followed, for a script if `ECMAScript` specification are followed, etc). This conformity is expected to keep at an ACCEPTABLE level, meaning at least fundamental principles.

Who to execute code review?

A code review **MUST** be done by **EXPERT** level members, both by a *developer* and a *software engineer*. This will maximize the process results and can give best information regarding **code generalization and reusability**.

Mob technique as code review technique

Mob technique as code review technique

SDEVEN recommends Mob technique ONLY for learning process when:

- junior members are used in coding process or
- when adopt a new standard, language, generally speaking a new "thing" and the knowledge should be transferred to some people

Mob programming means that *all required team members* are present in the same time in front of one screen. Or work remotely on a shared screen — that is my case.

First (the team or its leader) decide for a task (or issue *treated as next action), and when possible we rotate in* **driving sessions**. *A session means there is* a one driver — *one who types / clicks, and* one navigator — *which tells the driver what to do. The other team members keeps attention, and only when the navigator goes in a wrong direction, then interrupts**. Navigator navigates for 3 (max 5) minutes and then rotate.

Rotation means that driver now navigates — should know next step, navigator takes a rest, and others of the drives but less than half of them. And after 3 minutes another rotation, and again, …

This rotation style is intense. You have to keep attention all the time, otherwise you'll have to navigate in couple of minutes, and you'll have no idea how to navigate (by ***you*** is meant the team leader).To stay in shape we do regular breaks for bathroom / coffee, and of course a long break for a lunch. Goals of code view are fulfilled

Sharing knowledge is instant — every team member follows the mental process, and knows why was what done. Sharing responsibility in my opinion full — I take responsibility for everything that we produce as I can anytime say "I disagree" or "I have a better idea". Code structure is agreed by all team members, therefore is consistent and the best team members can do. Learning… is again instant, and intense. If the navigator is good, they'll not only call what to do, but also how to do it efficiently. I learn daily better software architecture, better testing strategies, how to use IDE efficiently, … just because navigators know (and share) pieces I'm missing.

Last update: August 13, 2023