Roadmap

1 Great steps

These are the planned steps:

- Confirm the assumption that the frequency of PMD Alerts is an accurate measure of the prevalence of kludges
- Confirm the assumption that kludges harm software development.
- Confirm the assumption that there is a game in which, in Nash equilibrium, a developer chooses a strategy in which he gets personal benefits while causing harm to the project, by making kludges.
- If all these assumptions are true, use mechanism design to devise how we can change the environment in a way that developers do not choose to make so much kludges anymore, increasing the quality of the project in the long term.
- Implement this mechanism building a plugin for a prominent CI tool, like Travis or Jenkins or GitLab.

2 Define Kludge

A kludge is code that

1. Partially fixes a bug or partially implements a feature.

The term partial can be understood as in partial function. A partial function is undefined for some elements in the formal domain, for example the square root function restricted to the integers: f(25) is defined, but f(26) is undefined. In terms of features, we can think about a developer calculating the point on which two lines cross and neglecting the case of parallel lines

2. Developer knows that the code is a only partial solution, with high probability.

Bruno must study technical debt papers to enrich the conceptual background.

3 RQ: Is the frequency of PMD Alerts an accurate measure of the prevalence of kludges?

We have to identify events with an intense PMD Alert introduction.

In order to do this we have to be able to categorize open, new and fixed PMD Alerts. This is the subject of another detailed report.

For each commit of group of contiguous commits, we can measure how much kludge introduction occurred. This kludge introduction must be normalized by the size of the commit. The measure may follow a formula like:

$$\frac{\#NewAlerts - \#FixedAlerts}{\#LoC}$$

With these events identified, including their location in the code and their moment in the time line, we can measure the correlation of these events with some evidence of kludge.

An evidence could be churns around the location of the event in subsequent commits. And this would be related with the statement 1 of the definition of kludge in Section 1. (It´s not clear for me, Bruno, if the chruns are an evidence of PMD alerts -> kludges -> software harm, that is, if it encompasses Sections 3 and 4.)

Other possible evidence could be survey and bag of words mining of issue, mailing lists, and commit messages showing evidence of kludge game. This could be related more with the statement 2 of the definition of kludge in Section 1

4 RQ: Do kludges harm software development?

We need some way to measure degradation after a heavy introduction of kludges

A drop in the popularity may not be a proper evidence.

The increment in the number of issues and bug fixed nor necessarily represent a degradation

Churn could be used here

5 The game

We have an idea related to the tragedy of commons and we know that in some situations the Nash Equilibrium would be in a state in which the developers make kludges, as in Gavidia-Calderon et al. (2020). We have to study an develop this subject much more

6 The intervention based on mechanism design

The paper Gavidia-Calderon et al. (2020) discusses the effect of an intervention: the addition of a single code reviewer

References

Gavidia-Calderon, Carlos, Federica Sarro, Mark Harman, and Earl T. Barr. 2020. "Game-Theoretic Analysis of Development Practices: Challenges and Opportunities." *Journal of Systems and Software* 159: 110424. https://doi.org/https://doi.org/10.1016/j.jss.2019.110424.