**Protocol to Select Incomplete Composites**

**for Pull Request Submission**

This protocol describes the requirements used for the manual selection of incomplete composites for the pull request submission in open source projects.

1. **Composites belonging to a single commit.**
   1. Recent studies [3][4] show that incomplete composites tend to apply in a single commit, even involving multiple code elements and refactoring types.
2. **Composites in the period from 10 October 2019 to 31 October 2019.**
   1. This period was chosen, since all refactorings were collected until 31 October 2019. So, it would be easier for developers to remember which composites were applied and why. Moreover, Bibiano *et* al. [3] note that composites often are applied within one to three months (about 70% of the composites). Thus, in this way, we can evaluate whether possible incomplete composites applied in the same commit are not completed by other composites in the same period of time.
3. **Composites consisting of *Extract Methods* and/or *Move Methods.***
   1. Studies [2][3][4] show that *Extract Methods* and *Move methods* are prone to do not remove *Code Smells.* Therefore, composites consisting of that refactoring types tend to be an incomplete composite to *Code Smell* removal.
4. **Composites applied to projects that have a frequent number of pull requests.**
   1. The projectis Dubbo, 120 pull requests. Values collected on 1 December 2019.
5. **Collect *Code Smells* that have not been removed from these commits in the code elements to which the composites have been applied.**
6. **Check that these code smells have not been removed in subsequent commits.**
   1. Until the day the pull request is sent.
7. **Check which code smell should be removed for each pull request candidate**
8. **Thinking about solutions to remove Code Smells.**
   1. The literature recommends composites to remove Code Smells [1][3][4]. In this study, we focus on Code Smells and Incomplete Composites that could be completed using recommendations from previous studies.

**References**

[1] Fowler, 1999, "Refactoring: Improving the Design of Existing Code".

[2] Cedrim, *et* al. Understanding the impact of refactoring on smells: A longitudinal study of 23 software projects. FSE’17.

[3] Bibiano *et* al. "A Quantitative Study on Characteristics and Effect of Batch Refactoring on Code Smells." ESEM'19.

[4] Sousa *et* al. Characterizing and Identifying Composite Refactorings: Concepts, Heuristics and Patterns. MSR'20.