Critique for CIDiff: Generate concise linked code differences

Ying Zhang

February 19, 2019

Summary: The exists code differencing methods are too fine-grained, and the code change summarization methods are too coarse-grained, to address this problem, this paper proposed and implemented a code differencing method named CLDIFF, this approach is able to generate the concise, linked presentation of code differencing. This approach includes three steps, it first creates ASTs for a pair of source code (unchanged and corresponding changed one) and pruning the unchanged AST elements; secondly, it generated fine-grained code differencing through GUMTREE [1], grouping the related code differencing at or above statement level to provide concise, high-level code differencing. Third, it defined five links and established the code change links between code differencing and pre-defined link.

They evaluated this tool with 12 high-stared open-source Java projects, and compared the performance with GUMTREE[1]. Through four research questions, they conclude the 1) CLDIFF implemented high accuracy for generating code differencing and establish the link; 2)it generated shorter edit script than GUMTREE; 3) the execution time was 72% shorter than GUMTREE; 4) this tool is more helpful based on human study and feedback collected from the participants.

Strength: Generating the concise code differencing (human understandable) is for code merging. Since the previous tool generates code differencing based on AST and too fine-grained, it is hard for developers to quickly get the code changes if they are not familiar with the AST structure. But the idea for grouping the in or higher statement-level code differencing then provide the code differencing is novel.

Weakness: For the evaluation part, 1)this paper didn't offer recall of the tool but just precision. Purely based on accuracy is not convinced (if the tool could only distinguish 10% of the code differencing, we can not conclude it perform well). 2) decide the tool is useful based on 10 participants who are graduate students from their school. (personally, I think this part of the result is not objective.) 3) the linked are limited by five pre-defined rules

Analysis: according to the snapshot of their visualization tool, the code differencing information provided to the developer is still hard for the human to read. The information primarily provides the change of the code, but lack of the motivation for code changing. For example, in Figure 5, the message "addMethodDeclaration" didn't provide any detail information about why make this change, it is still hard for developers to make code merging based on this information. In my opinion, it should allow programmers to add comments to the message or extract the related comment when showing the message. That will help the programmer to understand the changes better.

References

[1] Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. Fine-grained and accurate source code differencing. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 313–324. ACM, 2014.