New Abstract (198 Words):

Most of the changes in a software system are done by reusing existing codes which creates source code clones in the codebase. To maintain consistency in a software system, all these cloned co-changes must need to be changed together during software evolution. Detecting these cloned co-change candidates is essential for clone tracking in the codebase. Earlier studies showed that clone detection tools can be used to enhance the performance of finding those co-change candidates. Though there are several studies to evaluate the clone detection tools based on their performance in detecting cloned fragments, we found no study which compares different clone detection tools in the perspective of detecting cloned co-change candidates. In this study, we wanted to explore this dimension of code clone research. We used 12 different configurations of nine promising clone detection tools to identify cloned co-change candidates from eight C and Java-based subject systems and evaluated the performance of those clone detection tools in detecting cloned co-change fragments. Evaluated rank list and relevant analysis of obtained results provide important insights and guidelines about selecting the clone detection tools which can enrich a new dimension of code clone research in change impact analysis of software systems.

Current Abstract (258 Words):

Code reuse by copying and pasting from one place to another place in a codebase is a very common scenario in software development which is also one of the most typical reasons for introducing code clones. There is a huge availability of tools to detect such cloned fragments and a lot of studies have already been done for efficient clone detection. There are also several studies for evaluating those tools considering their clone detection effectiveness. Unfortunately, we find no study which compares different clone detection tools in the perspective of detecting cloned co-change candidates during software evolution. Detecting cloned co-change candidates is essential for clone tracking. In this study, we wanted to explore this dimension of code clone research. We used 12 different configurations of nine promising clone detection tools to identify cloned co-change candidates from eight C and Java-based subject systems and evaluated the performance of those clone detection tools in detecting the cloned co-change fragments. Our findings show that two configurations of CloneWorks for Type 3 clones (i.e., Type3Pattern and Type3Token) significantly perform very good, Deckard and CCFinder are in the following order based on the performance. Summarizing the overall findings show that a good clone detector may not perform well in detecting cloned co-change candidates. The amount of unique lines covered by clone fragments, the number of detected clone fragments, source file processing mechanism, and the types of detected clones play important roles in detecting cloned co-change candidates. The findings of this study can enrich a new dimension of code clone research in software engineering.